TM687
2628.000

Fermilab

THE 15 FOOT BUBBLE CHAMBER ONLINE COMPUTER SYSTEM

W. M. Smart
October, 1976

ABSTRACT

The online computer system in use at the 15' bubble chamber is described, with emphasis on the program. The system is used to log and display data about the bubble chamber and its support systems, and has been very useful in improving both the efficiency of the chamber operation and the physics value of the bubble chamber pictures. This note is sufficiently detailed to serve as a useful guide to the online program. Experiences and techniques of possible value for other small computer projects are given. Paragraphs of more general interest have been flagged for the reader who is uninterested in program details.

TABLE OF CONTENTS

THE 15 FOOT BUBBLE CHAMBER ONLINE COMPUTER SYSTEM

I. INTRODUCTION

* This note describes the online computer system in use for data logging at the Fermilab 15' bubble chamber. It is intended to serve two audiences: those wishing to obtain a general idea of what the system does; and to give detailed descriptions of the program so that an experienced PDP-11 assembly language programmer, by using this note and the actual program listings, could have enough information to either modify this program or borrow some of the program's features for another project. To spare the former group unnecessary detail, I have marked the paragraphs which are relevent to a general overview of the computer system with a * in the left margin. Sections I through IV, XII and XIII are the most important for this overview.

* Most of the note is concerned with what the online program does and how it does it. Only a brief description of the computer hardware, devices interfaced to the computer, systems programs supplied by Digital Equipment Corporation and Fermilab computer group programs will be given, but most of these are covered by the references appearing at the end of the paper. Operating instructions and commands to the outline program also are not given here, they are kept in a notebook in the bubble chamber control room.

II. PURPOSE OF THE ONLINE COMPUTER SYSTEM

* The primary purpose of the computer system is to gather data about the 15' bubble chamber, store it for later use, and display selected data to the bubble chamber operators. Most of this data is of interest only to the bubble chamber operators, but some data is expected to be of interest to the physicists running an experiment in the 15' bubble chamber. Such data includes: the date, time, roll number, frame number, bubble chamber magnet current, chamber temperature, chamber pressure and pressure drop, and number of hadron beam particles entering the chamber; all of which are recorded on magnetic tape each time the chamber is pulsed.

* Data of interest to the operators includes more detailed information about the bubble chamber itself: temperatures and pressures at several points inside the chamber, at the piston rings and under the piston, cooling loop parameters, etc.; and information about the bubble chamber support systems, such as the superconducting magnet and helium liquifier, hydrogen refrigerator, expansion system, and gas and liquid storage tanks. Data of interest in these support systems includes temperatures, pressures, flow rates, liquid levels, valve settings, etc. Having this information available enables the operators to run the chamber more efficiently and economically.

* Using a computer to log this data has several distinct advantages compared to logging it manually. Sets of readings can be taken more frequently and on a regular time basis. Such reading continues automatically, even when all the operating crew is involved with some chamber problem. Data recorded during such

problem times are frequently very important in understanding the malfunction and would probably not be taken without the computer because the crew is busy working on the problem. Computer readings are usually more accurate and the computer can quickly convert readings to the appropriate physical units. The computer can average several related readings taken at (essentially) the same time, for example, the several vapor pressure thermometers inside the chamber volume can be averaged to give the average chamber temperature. Averages can also be made of all the readings of a particular piece of data taken in specified time periods. Another job that the computer can do well is to calculate the rate of change of a particular reading with time. This information can be extremely useful, one example is the cooling rate of the bubble chamber windows and chamber body during chamber cooldown from room to liquid hydrogen temperatures. Too high a cooldown rate could damage the glass bubble chamber windows. Another example, see Figure 1, is the level in the liquid helium storage dewar. During magnet operation, liquid helium is continuously added to the dewar by a liquifier and liquid from the dewar is transferred to the magnet periodically when it is needed. Reading the helium level in the dewar shows only how much liquid is in the dewar, the rate of change of the level tells you how much excess liquid is being made or, if negative, how long the magnet can be run under the present conditions. The helium liquifier can be tuned up using this rate information. Studying many level readings taken over a period of time will give the period and quantity of the batch liquid transfers to the magnet. The computer can store a large amount of data, some of it up to three weeks old, on the disk which is then immediately

available for the operator to study. Data can be stored on magnetic tape for indefinite periods and retrieved offline on a large computer. Such magnetic tapes are far less bulky then recording the same amount of information on paper. Finally, the computer can make listings of selected data, including data acquired up to three weeks earlier, and make graphs of such data at the operator's request.

* In short, the computer can save operators time by recording, analyzing, and displaying data and record it more frequently, accurately, regularly and usefully than it could be done by hand. In my experience at the 15' bubble chamber, there has been a real need for data which is both accurate and quickly available to improve both bubble chamber track quality and the efficiency of bubble chamber operations.

* At the present time, the computer has no control functions over the bubble chamber. There are several reasons for this. Early in the design of the computer system, it was decided not to make operation of the chamber dependent upon the computer to the extent of requiring the computer system to be up before the chamber could run. Almost all simple routine control operations at the 15' bubble chamber are done by commercial air system controllers. More complicated control operations have long cycle times and are easily handled by the operators once they have accurate data on which to base their decisions, so the first priority for the computer system has been to provide that data. Now that the data logging features of the computer system are almost completely implemented, some control functions for the computer may be advisable and these are discussed in Section VII.

III. HARDWARE

* The basic hardware used for the 15' bubble chamber on line system is a rather typical Bison system provided by the Fermilab Computing Department. The computer is a Digital Equipment Corporation (DEC) PDP-11/20 with extended arithmetic element (EAE) and 28 K (K=1024) words of memory, the maximum possible on a PDP-11/20. Major DEC supplied peripherals include a 1.2 million word cartridge disk, 800 bits per inch 9 track magnetic tape unit, dual Dectape unit and a 30 character per second Decwriter terminal. The Computing Department also supplied a memory display scope, 600 lines per minute printer/plotter, Bison interrupt and gate control unit, and CAMAC branch driver, together with the necessary interfaces and controllers. A more detailed list of the hardware supplied by the Fermilab Computing Department is given in Table I.

* The PDP-11/20 is a small computer with 16 bit words and a typical instruction execution time of 5 microseconds. It has powerful input-output features which enable it to handle data transfers very quickly and with a minimum delay to the computing which occurs in parallel. Single precision integer arithmetic operations, including multiplication and division (with the EAE), are quickly done and fairly easy to program, but multiple precision operations take longer and are more difficult to program. Floating point arithmetic hardware is not available for a PDP-11/20 and floating point software routines take considerable memory space and are rather slow. The speed of the computer is more than ample for almost all of the demands at the bubble chamber. The exception is in the analysis of data during the chamber pulse, as described in Section VI, but with careful programming

the computer execution speed is adequate even for this job. The major shortcoming of the PDP-11/20 has been that the memory is limited to 28 K words, and considerable programming effort has been necessary to fit the program into this available memory, as described in the next section. More information about the computer can be found in DEC supplied literature[1].

* With the computer hardware as outlined above, there are four places where data, programs, etc., can be stored and these are given in Table II in order of access time, with the fastest device first. Both the disk and the Dectapes are hardware organized into 256 word blocks, so the number of such blocks is given for the other storage devices on the bubble chamber system for comparison. Since the hardware block size on the disk is 256 words, this block size is used throughout the online program for data storage. General data buffers in memory are 256 words long and data is written on the magnetic tape in 256 word records. The first word of each block contains an identifying code, while the next three words contain the date and time the data was created in the usual PDP-11 format. Usually, the remaining 252 words contain data, see Section VII for details. The disk is used by the online program to hold a large number of data, control, and constants blocks which can be recalled in a very short time (less than .125 seconds per block). The magnetic tape is used to log data for later offline analysis. To obtain more disk space for the online program, a second disk cartridge is used to hold the complete set of disk operating system (DOS) programs and for program development. The Dectapes are not used directly by the online program, but they are used to hold backup program source

files and to transfer the online program load module between disk cartridges.

* In addition to the hardware supplied by the Computing Department, we have added additional peripherals to tailor the system to meet the needs at the bubble chamber. These include a CRT terminal (identical to those used on the Beam Line MAC systems), a fast (25 microseconds per data point) Datel analog to digital converter (A/D) with 128 multiplexed differential inputs (8 of which are sample and hold) and 8 digital to analog outputs, a digital voltmeter (DVM, 0.5 seconds per data point) with 64 differential inputs, a Scanivalve air signal scanner (0.167 seconds per data point) with 64 inputs, and a CAMAC crate which contains modules for digital input and output as well as 8 channels of high speed scalars. The fast A/D is interfaced to the PDP-11 unibus for both programmed and direct memory data transfers. The CRT terminal, DVM, and CAMAC crate are interfaced to the unibus for only programmed data transfers. The air signal scanner is not connected directly to the unibus but the addressing commands and address read back are transmitted through the CAMAC crate, the analog pressure transducer data signal input goes to the fast A/D, and the data ready interrupt is handled by the Bison interrupt and gate control unit. Two bubble chamber thermocouple read out systems, each for 100 inputs, are connected to the computer via the CAMAC crate. Each system is based on a special DVM which converts the thermocouple reading into temperature in degrees Kelvin. Normally, the computer controls each thermocouple system, although manual operation is also possible in case the computer is down. In addition to other digital inputs, the CAMAC crate also handles the output from the computer to a 16 digit display of key chamber expansion parameters.

* These special devices are listed in Table III. More information

on the peripherals can be found in reference 2, from the manufacturer, or in the manuals on file at the 15' bubble chamber.

IV. PROGRAM INTRODUCTION

* One of the most important considerations for the bubble chamber computer system is that it should require as little operator action as possible. Many bubble chamber operators have had little experience with computers and frequently they are almost completely occupied with the operation of the bubble chamber itself. To meet this requirement, the program has been written to be as easy as possible to restart after a program bomb-out or a power failure, once it has been restarted all data read in and logging is automatic, and the commands to display data, alter limit checks, etc., have been kept as simple as possible.

* To start the computer, the operator follows the instructions for starting any PDP-11 (using DOS, the disk operating system[3]), which include entering the current date and time and typing a few commands on the Decwriter. Once the program is started, the operator must tell it where to start logging data on the magnetic tape, which usually requires only one or two commands. All data read in and logging will now begin and no further action is required of the operator. The latest constants and control tables needed by the program are stored on the disk, so even recent changes are already available and do not need to be re-entered by the operator. Also, the data logged on the disk when the program was running previously are immediately available for display on operator command.

* The read in of general data occurs automatically at predetermined times using the internal computer clock. After the data has been read in and processed, it is stored on the disk in a place determined by the time when it was read in. The new data is written over older data on the disk, so that the most recent data is available for the

optional displays. The amount of recent data kept on the disk before it is overwritten depends on the type of data. This varies from 4 hours worth (all fast data, see Sections V and VII) to 3 weeks worth of the data averages. The data is also logged on magnetic tape for later offline analysis. Every time the chamber is pulsed, the computer is interrupted to read in data concerning that pulse. After processing,this data is also stored on the disk and logged on the magnetic tape. In this case, the last 1024 pulses are kept on the disk as well as the average values which are saved for three weeks.

* In most cases, display of information to the bubble chamber operator is optional and requires that a command be typed at the control CRT keyboard. Because data logging is automatic and because a considerable amount of data is stored on the disk, the operator need not look at the data at the time it is read in, but has the freedom to look at it several hours or even several weeks later. The operator can choose from several ways of displaying the data. He may select a list of many different data points at one selected time to be output temporarily on the memory scope, or permanently on the line printer. Or, he can select a few data points (up to 7) and output a list either on the memory scope or line printer, of the values at many successive times. It is also possible to display, on the memory scope, a plot of any data point vs. time, four small plots each of a different data point vs. time, or a plot of one data point vs. another data point using many pairs of readings, each taken at a different time. These plots can be copied from the memory scope to the line printer/plotter, using the Bison hard copy facility[4]. Better

resolution plots of any data point vs. time may be made directly on the line printer using the Bison routine PLOTB[5].

* Another feature of the program is the ability to make changes to the constants and control tables stored on the disk. The CRT control terminal is particularly useful for this, because the program has been written so that old values of the constants are displayed on a line and the operator need only run the cursor over to the value(s) he wishes to change and type in the changes. (Similar to the edit facility used in some cases on the beam line MAC systems.) This is especially useful for modifying the limit checks (similar to watch lists on the MAC systems) which may be frequently changed by the bubble chamber operators. In this case, the lower limit, current value, and upper limit are displayed on a line and the operator can change the limit(s) while having the current value right in front of him.

* The online program has been developed using the DOS (Disk Operating System) provided by DEC (Digital Equipment Corporation). DOS provides the tools necessary to create and modify programs, translate them to machine language and run these programs. Additional tools are provided to assist in these steps, all of which are described in the literature which DEC provides[3]. When the online program is running, a small portion of the DOS monitor (2655 words) remains in memory to assist in input-output operations, provide error diagnostics, assist in debugging programs, handle resetting the computer clock, etc. It also provides an easy method of returning to the DOS system when execution of the online program is terminated by operator command.

* The online program also uses the BSX multi-task supervisor[6],

developed here at Fermilab, to allocate the control of the computer central processor (CPU) to the various tasks which make up the online program.

Each task functions logically as a separate subprogram. Usually, each task is waiting until an event variable (word in memory) becomes non-zero (an orwait on a list of two or more event variables is also possible). In this wait state, the task neither requires nor receives control of the CPU. The event variable can be set non-zero either by another task or by an interrupt service routine which receives control of the CPU as a result of a hardware interrupt. When a task's event variable is set, BSX gives control of the CPU to that task, provided no task with higher priority also requires the CPU. Once a task receives control of the CPU, it retains it until a hardware interrupt occurs or the task reaches the point where it must wait on the same (or another) event variable for some action, external to the task, to occur. This later case may occur, for example, when the task cannot proceed until a certain amount of time has elapsed or a requested output is completed, or it may occur when the task has finished its job and must wait until new work is required of it. In the case of a hardware interrupt, the registers being used by the task are saved and the CPU is then used for the action specified by the interrupt. When the interrupt action is over, BSX checks the higher priority tasks and gives control to the highest priority task needing the CPU. Once all the higher priority tasks are satisfied, BSX restores the registers and continues the execution of the interrupted task at the point where it was interrupted.

* BSX thus provides the means to do "first things first" which is

vital to any real time program, while still allowing low priority jobs to be completed as soon as possible. BSX also handles the input-output operations to the standard Bison devices: the magnetic tape, disk, line printer, memory scope, DEC writer (output only), and the control CRT terminal. The first four of these use the standard DOS device drivers which require 1353 words of memory. I have modified the Decwriter driver KBI0HT[7], written here at Fermilab, for the control CRT terminal used for the online program. A cut down version of KBI0HT serves to drive the Decwriter for output only. These two drivers require 830 words of memory. The BSX supervisor, including the task table, requires 1515 words of memory.

* The online program currently contains 18 tasks, which are supervised by BSX. Seven of these are required for the six standard BISON devices listed in the last paragraph. (The control CRT terminal requires two tasks, one for keyboard input and the other to output on the CRT screen.) These tasks are described in the BISON program notes[6]. Five tasks are used for the reading, processing and logging of general data and are described in Section V. Three tasks are used to log and display the data gathered during the chamber pulse, see Sections VI, VII and VIII. The clock task, which is entered every 1/2 second as a result of an interrupt from the internal programmable clock[8], is responsible for initiating the read in of general data and setting event variables after an interval of time as requested by the other tasks in the program, see Section X. The command task waits for the operator to enter a command on the keyboard, interprets the command using the BISON subroutine CICONV[9], and then carries out the action requested by the command, see

Sections XIII and IX. In the case of certain commands to list or display data, which may take considerable time to complete, the command task transfers the request to the low priority list task and is then available to accept another command. The list task is described in Section VIII.

* Outside this task structure of the outline program are several interrupt service routines which are executed as a result of hardware interrupts. Most of these interrupts indicate that an input-output operation has been completed and the service routines are rather short and serve to notify a task that the operation has been completed and perhaps initiate further I/O operations. Two exceptions are: the clock interrupt service, CKINTR[8], which updates the current date and time words stored in the monitor and starts the clock task described above; and the pulse interrupt service routine, described in Section VI, which reads in and processes data during the bubble chamber pulse.

* A major problem when trying to do a big job with a small computer, such as a PDP-11/20, is to fit the program into the available memory. The remainder of this Section will outline the steps that have been taken to reduce memory requirements of the bubble chamber online program and some additional information can be found in Section X. For the last year, every major addition to the program has required considerable effort to reduce the memory required for the previous version of the program, before the new feature could be added.

To use the available memory as efficiently as possible, the online program is written entirely in Macro assembly language. Assembly language generally creates one computer instruction for

each line of code, while a higher language, such as Fortran, will generate many computer instructions for each line of code. Assembly language, since it corresponds closely to the actual hardware of the computer, allows the programmer to make full use of that hardware to reduce both memory requirements and execution time for a given job. Fortran, especially for the PDP-11 computer, is slower and uses much more memory than assembly language. Disadvantages of assembly language programs are: since it coupled closely to the actual hardware of the computer, it is almost impossible to transfer programs between different types of computers; it takes several months for a programmer, even if he is proficient in Fortran programming, to learn to write assembly language programs reasonably well; and finally, even an experienced programmer takes far longer to generate and debug programs in assembly language than in Fortran. (One estimate is that a good assembly language programmer averages only three to five instructions per hour to generate and completely debug a program.) In spite of these disadvantages, the need to save memory has forced the use of assembly language programs exclusively in the bubble chamber online program. Some Fortran was used for small independent hardware check out programs, and in early versions of the online program, before it had the current capabilities which are described in this note.

Another restriction imposed by memory size and hardware limitations has been to only store data as single precision integers. With the 16 bit word size on the PDP-11, this allows a range of -32,768 to +32,767. While this range contains ample precision for almost all bubble chamber data, it does not lend itself directly to convenient display of the information. Most of us are more

familiar with a thermocouple temperature expressed as 123.4 degrees Kelvin rather than just the integer 1234, and this becomes more confusing if the rate of change of that thermocouple is expressed as 10 rather than 1.0 deg/hr. The Bison output formatter program FMTPUT[10], was modified to insert a decimal point in such data before it is output. At the same time, the eight largest negative numbers were reserved as error codes and the formatter modified to print these out as *0 through *7. These error codes tell both the operator and the program that the data is not present and that the value should not be treated as valid data. Using different error codes gives the operator an indication as to why the data is not present.

Numerical operations, such as converting the raw (as read in) data to meaningful (inches, PSIA, etc.) units, calculating averages, standard deviations, etc., have been coded using integer arithmetic. Using the available software floating point arithmetic routines, although easier to program, would have required more memory space.

The bubble chamber data which is available to the operator is stored in 3584 blocks (917,504 words) on the disk. Clearly, it would be impossible to store more than a very small percentage of this data in the 112 blocks of memory available. Using the disk to store data in this way, of course, required writing assembly language programs to store data in the proper place on the disk and to retrieve it on command.

A sizable number of control and constants blocks are also stored on the disk and are used by the program to control its operations, especially to systematically process the general data, as described in Section V. Other disk blocks are used to hold temporary results,

such as the partial sums needed to calculate means and standard deviations every two hours. Currently, 96 such blocks are used, which are equivalent to a sizable fraction of the available memory. Such blocks are read into memory only when they are needed.

Another group of blocks on the disk is used to hold list specifications. These are equivalent to the WRITE and FORMAT statements that one would use in Fortran to output a list of variables. Putting these specifications on the disk allows for the possibility of a large number of lists, each with considerable titles to clearly show what data are being presented. The practical limit to the number of lists possible is the time and patience of the programmer in generating the blocks of list specifications. Currently, 15 lists are implemented and 50 are possible, but this limit could be easily increased if necessary. Twenty-eight additional blocks on the disk are used to hold these list specifications.

Whenever any of this information (data, control and constants, temporary results, or list specifications) stored on the disk is needed, it is read into a general buffer which has been reserved by the task needing the information. The task, after using the information, releases the general buffer so it can be used by other tasks. Currently, about 13 general buffers, each 256 words long, are needed for the online program to function smoothly. Most of these general buffers are generated in the remaining free memory just after the online program execution is started. Before these buffers are assigned, about 500 words of free space are needed for a check to be sure that the proper disk cartridge for the online program is in the disk unit. The online debugging system program

(ODT)[3], which is very useful in getting the program to work properly after modifications, is always linked and loaded with the online program. However, if it is not going to be used, the memory space it occupies is assigned as general buffers during the online program start-up procedure. Finally, the code for the start-up procedure, see Section X, is all in one area, and this area is also used as a general buffer after the program start-up phase is complete. Currently, there is enough space for 17 general buffers, if ODT is not needed.

* Aside from increasing the amount of information storage available far beyond the actual memory size, this extensive use of the disk has another benefit. The storage on the disk is permanent and all the latest information is immediately available when the online program is started up after a power failure, program bombout, or other interruptions. Changing control blocks, constants blocks, or list specifications on the disk can be done (see Section IX) easily and quickly while the online program is running. Changing the online program itself requires 10 minutes or longer and requires stopping the execution of the online program (including data read in and logging) for that period.

Much of the online program is needed to handle the read in, processing, and logging of data and this code must be in memory at all times while the program is running. The portions of the program which service the optional operator requests for lists, summaries, plots, and modification of disk blocks are only needed when these requests are made and have been overlayed[3]. Currently, there are three overlay segments which share an area in memory about 4100 words long. If additional display features are desired, other

overlay segments of similar length can easily be added without using more memory. Use of the overlay feature allowed the program to be increased by about 7700 words (the length of the two shorter overlay segments) while only increasing the program memory requirement by about 400 words (the overhead needed to handle the overlaying).

* Each of the steps outlined above has enabled the online program to accomplish more jobs without increasing the actual memory required. Some of the steps have had additional benefits, but all the steps have required considerable programming effort which is not at all apparent to someone unfamiliar with small computers. A larger, more powerful, and more expensive computer would reduce the amount of such programming effort, but in this case, the cost of the software was less than that for more powerful hardware. Supervisors should realize, however, that a considerable programming effort must be invested to effectively use a small computer system. In this case, about 15 full time months of my time was required for the program, spread out over the last two years.

V. READING AND PROCESSING OF GENERAL DATA

* This section describes the handling of general data concerning the bubble chamber and its support systems. This data is slowly varying with time and uncertainties of a few seconds in the time at which it is read make little practical difference. Taking advantage of this fact, the data is read in and processed in large blocks identified by the date and time the read in process started and analyzed by subroutines which operate on all data in the block in a systematic way. The control and constants blocks needed by these subroutines to read in and analyze the data are stored on the disk and read into general purpose buffers only when they are needed. Two general data blocks are handled in this manner. The read in of the fast (slow) data block is initiated every 1/64 (1/16) hour by the clock task. Each data block is organized with an identifying word first, then a date word and two time words. The read in (which may take considerable time) and initial analysis of each block is handled by a separate task, so that these operations can occur in parallel. The final analysis is done sequentially by a third task, which requires four general buffers and 1368 words of memory. The read in tasks require 3675 words of memory (including input device drivers, current values of the data for both blocks and status bytes for each data word). When it is active, each read in task also requires two general buffers. Aside from saving memory, keeping the control and constants blocks on the disk allows permanent changes to be made in them while the online program is running. This means, for example, that the read in of new pieces of data can be added to either the slow or fast data blocks without interrupting the read in and processing of all the other data by

the online program.

The data for these two blocks can come from any of the devices listed in Table IV. The two thermocouple (TC) systems each contain a special digital voltmeter which reads the rather low TC voltages and converts to temperatures in degrees Kelvin. The setting time for reading each TC is typically three seconds and there are currently about 60 TC's on each system. The gold chromel TC's are installed on the bubble chamber proper and the copper constantan TC's, for historical reasons, are installed on the superconducting magnet, hydrogen refrigerator, and the room temperature optics on the chamber. The digital volt meter (DVM) reads general data with a settling time of 0.5 second. The high order octal digit of the subaddress for the DVM determines one of four ranges as given in Table V. The Fast A/D (Analog to Digital) converter reads one general data point in about 25 μseconds (not counting program overhead time), with a least count of about 5 MV, and full scale range of ±10 volts. The Scanivalve reads sixty-four 3 to 15 psi air logic signals at the rate of six per second. These air logic signals measure, for example, the amount various control valves are open, pressures, flow rates, and liquid levels. Unlike all the other devices here, which are random access, the Scanivalve is sequential, so the 64 air logic signals are put sequentially into 64 words in the fast data block. Digital input is handled by sixteen 24 bit words in the CAMAC crate. Because of the mismatch between the 24 bit CAMAC words and the 16 bit PDP-11 words and the fact that digital inputs are not always exactly 16 bits long, considerable shuffling around must be done by the CAMAC device handler and the exact number of data points that can be handled by

the available input cannot be given. Currently, the first eight 24 bit CAMAC words handle eight data points (magnet current, magnet voltage, magnet power supply voltage, precision chamber static pressure, and roll and frame for each of the two experiments) as well as the data from the two TC systems and the subaddress read back for the TC systems and the Scanivalve.

The rate of 16 reads an hour for the slow data block was chosen to match the time required to read in all the TC's in each of the two TC systems. The rate of 64 reads an hour for the fast data block could be doubled if there was a real need for data that often and more disk storage was available to keep the data for a reasonable amount of time. The present rate also allows time to read all 64 DVM data points into the fast data block, if desired.

At the proper time to start a read in, the clock task sets the appropriate event variable. As soon as the central processor is available, the requested task starts the read in procedure. First, two general purpose buffers are reserved. One will be used to hold the raw data for this read. The correct ID code and current date and time are put into the first four words of this buffer. The read in control list is read from the disk into the other buffer. The four ID, date and time words contain the date and time when the read in control list was last modified; the remaining 252 words control what piece of data is read into the corresponding location in the raw data buffer. These words contain the device number in the high byte (high order eight bits) from Table IV, and the subaddress in the low byte. The device driver indicated by the first of these control words is now called with the subaddress and location for the data as arguments. Also included as an argument is a busy

flag; the device driver will return zero here, if it has accepted the input request or, if the driver is busy, it will return the address of an event variable that will be set nonzero when the driver is free. Also included in the arguments, is the address of a counter of the number of data points which have been requested but not yet read in. The driver must increment this counter when it accepts a request and decrement it when that data has been read in. When this counter becomes zero, the driver sets an event variable to show that no more data is outstanding; the address of this event variable is also an argument.

The action of the drivers for the six devices listed in Table IV is rather different. The drivers for the TC systems run as separate tasks, because they actually read the requested TC every 0.5 seconds, apply a test to see if the reading has settled and then wait, read and test again if the reading is still changing. If a time limit of 7.5 seconds elapses without convergence, or the meter overloads, an error code is put in place of the data. The DVM driver sets the requested address and then returns. After the 0.5 second for settling has elapsed, the DVM sends an interrupt and the interrupt service routine reads the data and sets the flag to show the DVM is no longer busy. The scanivalve driver sets a counter for the 64 subaddresses, orders the scanivalve to the 0 address (home position) and returns. When address 0 is ready to read, the scanivalve sends an interrupt and the interrupt service routine reads the data, applies a calibration (for the pressure transducer in the scanivalve), and stores the calibrated data in the next location in the raw data buffer. If all 64 data points have not been read in, the scanivalve

is stepped to the next subaddress and the next interrupt waited for; if all data has been read, the flag is set to show that the scanivalve is no longer busy. The fast A/D and CAMAC inputs require only a few tens of microseconds to read a data point, so these two device drivers immediately read the data and then return.

The read in tasks contain five different scanners of the read in control list. The general scanner reads in devices numbered five and higher and turns on four special scanners, one for each of the first four devices in Table IV. Each of these scanners looks through the read in control list for the next entry requesting the read in of its device(s). The scanner then requests the device driver to read the desired data. If the request is accepted, the scanner then continues scanning the control list; if the request is refused (device busy), the scanner waits until the device driver is free and reissues the request. When each scanner finishes the read in control list, it is marked done. When all scanners are done and the outstanding data counter goes to zero, indicating that all the requested raw data has been read in, the task goes on to the next step. The special scanners are needed to allow the first four devices, all of which are DVM based and hence, rather slow, to be reading data in parallel. This reduces the read in time to the minimum possible, without putting any restrictions on the order of the requests in the read in control list. If some problem with the read in of a data point occurs or there was no request to read any data in (zero device number in the read in control list), the appropriate error code is put in place of the data.

The next step, once all the data has been read in, is to convert the raw data to data in the appropriate physical units. First a

flag is set to show that the data in the permanent 256 word memory block is changing. Next the first of four constants blocks is read from the disk into the second general buffer reserved by the task. The constants blocks contain four entries for each of the 252 data points. The first entry is a control word, which is followed by three calibration constants; b, s, and a:

$$(\text{physical data}) = (\text{raw data}) * b * 2^{s-16} + a \qquad 1)$$

Since both data and constants are stored as single precision PDP-11 integers (15 bits plus one sign bit), the s acts as a sort of poor man's floating point to increase the accuracy of the calibration in equation 1. Bit 0 of the control word, is set if this calibration is to be applied. The remaining bits of the control word, if non-zero, indicate a subroutine to call to do further operations on the data. Currently, the only such subroutine is one which adjusts the reading of control values (Scanivalve input) to be between 0 and 100% open. The calibrated data is then put into the permanently assigned block in memory which contains the latest data points in physical units. Once the first fourth (63) of the data points have been processed, the second constants block is read into the general buffer, used to process the next 63 data points in the block, and so on. If the error code is encountered,in place of the data, that error code is transferred to the final buffer without any modification. If the result of equation 1 overflows, an error code indicating that the data was too big is entered in the final buffer in place of the data.

Next, any desired derived quantities, such as averages, differences, etc. are calculated and put in the final buffer. Another possibility

is to calculate the short term average of a key pulse parameter. The sum of the data and the number of entries are stored in memory after each pulse by a high priority task. Now the average is calculated, the input memory words cleared for the next interval, and the average stored in the final buffer. The specifications for this step are stored in one or more blocks on the disk, and the first such block is now read into the general buffer. Unlike previous steps which operated on all data points with very similar operations, here only specific data points are used in operations that can be very different. The control block has the usual first four words, but the next work is a link (the block number) to the next control block. The last (or only) control block will have 0 in this word. The operations to be performed are each specified by a string of words in the control block; each string is terminated by a zero. The first word in each string specifies the subroutine number that the derived quantities processor will use. Next is a word that specifies where the data is to be stored in the final data buffer, followed by words specifying the input. These words can be constants, specify data that has just been put into this final buffer or the latest values of data in the other final data buffer. This last possibility is one of the major reasons that the latest values of the data for both the fast and slow data blocks are kept permanently in memory. A second zero, after the one terminating the string, ends action on the current control block. If the link to the next block is nonzero, that block will be read in and used; if it is zero, this step is complete.

Next, the flag is set to show that the data in the final buffer is no longer changing, the two general buffers are released, and

an event variable is set to signal the task which does further analysis of the data that a new final data buffer is ready to process. The final data is written on the disk in the area for all data of this (fast or slow) type and, if the read request was at an even ¼ hour, the data is also written on the disk in the area for quarter hour data of this type. Finally, if the magnetic tape logging is active, the data is also written out as the next record on the tape.

If a read in overrun occurs, i.e., the time to start the next read comes up while the task is still in the raw data read in step, the old read in is terminated, the error code indicating overrun is put in the unread data locations, the task waits several seconds for any outstanding requests to the device drivers to be completed, and the old data is processed by the remaining steps of the task. The read in of the new data then begins.

The read in tasks for both the slow and fast data blocks are virtually identical, in fact, they both share some re-entrant code. Of course, they use different control blocks from the disk and read different sets of data as a result. Some care must be exercised in setting up the read lists for these control blocks to be sure that no device is requested to read more data points than it can in the available lime.

The data processing task is now activated by the event variable which was set by the read in task after all new data was in the permanent memory buffer. This task completes the analysis of the new data. First, it reserves two general buffers for its own use. Its first job is to add the new data to some partial sums which are stored in six blocks on the disk. These partial sums are used

every two hours to calculate the mean and standard deviation of each point for all the reads during this period. The partial sums require six words for each data point. The number of reads so far in the two hour period is stored in the first word (single precision), the sum of the data is stored in the next two words (double precision), and the sum of each datum squared is stored in the last three words (triple precision). The sums are updated by reading the first of the six partial sum blocks into a general buffer, adding the current data for the first 42 data points to the 6x42 partial sum data words in the first block, and then writing the new information back on the disk. This process is repeated for the remaining five blocks.

The next step is to calculate the rate at which all the data points are changing with time. These rates are always expressed in units per hour and are calculated with respect to the data read 15 minutes and 60 minutes ago. First, the data from 15 (60) minutes ago is read into one of the general buffers. The data and time words of this old data block are then checked to be sure that the block contains data read 15 (60) minutes ago, if it doesn't, a flag is set to show that there is no data for these rates. This could occur, for example, if the online program had just been started after having not been run for a significant period of time. The value of the current data minus the old data (times four if calculating a 15 minute rate) is then stored in place of the old data. If either the current or the old data is missing (i.e., an error code), the error code for no data is stored instead. After these calculations are complete, one of the general buffers will contain the rates based on data read 15 minutes ago, and the other will contain the

rates based on data read 60 minutes ago.

* The program contains a special provision to automatically print a list of the gold chromel TC values and rates on the memory scope every time the slow data buffer is processed. This data has become a virtual necessity for the bubble chamber operator to correctly monitor and control the bubble chamber cooldown, particularly in keeping the cooldown rates of the glass bubble chamber windows within safe limits. Before the online computer system was available, one operator was kept busy full-time watching no more than three or four TC's, calculating the cooldown rates by hand, and controlling the cooldown valves. A second operator was kept busy almost full-time reading and recording all 60 TC's every half hour. With the online computer system, the values and rates for all 60 TC's, as well as averages of groups of TC's, are displayed 16 times an hour. One operator, spending only part of his time, can easily control the chamber cooldown and the danger of excessive cooldown rates on the glass windows has been reduced considerably.

This list is printed out on the memory scope now by the data processing task using the values stored in the permanent data block in memory and the rates stored in the two general buffers. The print of this list everytime the slow data is read can be turned off when it is not needed.

* The next step is to check all the new data and give a warning message and (optionally) an alarm, if it is outside limits which have been set by the bubble chamber operator. These checks may be on the current value, the 15 minute rates, or the 60 minute rates. Four sets of limit checks for each of the 252 points in both the

fast and slow data blocks are possible, but only a small subset of these are expected to be active at any one time.

The control and limits used in these checks are stored in nine blocks on the disk. Nine words are used for each data point. The first of these is a control word which uses four bits for each of the limit checks. Two of these bits indicate that the limit check is turned off, or is to be made on the value, 15 minute rate, or 60 minute rate. The third bit, if set, will cause the alarm to be sounded if the check is outside limits. The fourth bit, if set, indicates that the operator has acknowledged the out of limit condition for this check, and no further tests are made for it until the bit is reset. The remaining eight words are the low and high limits for each of the four checks.

The limit checking can be turned off by a command from the keyboard; however, if this is done, the alarm indication remains on continuously to remind the operator that the limit checks are disabled. If the limit checks are active, two more general buffers are reserved. One is used to sequentially hold the nine disk blocks containing the control and limit data. The data points are checked sequentially, skipping any checks which are turned off, acknowledged, or which refer to a value or rate containing an error code. When a check shows data outside of limits, the alarm control bit is tested and the alarm indication made if the bit is set. The identification block for this data point is then read into the fourth general buffer (unless it is already there because of a previous limit check). The data point name, number, units, and format is taken from this identification block for the message. The message is always printed out on the decwriter for a permanent record and

includes the value (or rate) which was out of limits, the limits, S or F (for slow or fast data block), the data point number, limit check number, date, and time, in addition to the above. If the check caused an alarm, a * appears next to the value. If the CRT screen is not being used for some other purpose, the message will also appear there, except that a blinking value replaces the * if an alarm resulted. After the message output is complete, the alarm indication is removed. When all limit checks have been made, the analysis of the new data is complete and the general buffers reserved for this task are released. The task then goes to an orwait until more work is required of it.

If the task receives a second request for action while it is printing out limit check messages, it prints a message that it has aborted the list of messages and then goes to the new request.

Aside from completing the analysis of new data in the fast and slow data blocks, this task has the third job of calculating the means and standard deviations for pulse, fast, and slow data every two hours. This requires three general buffers, one for holding a block of the partial sums of the data read in the last two hours, and one each for the output means and standard deviations. The equations used are:

$$\bar{x} = \frac{1}{N} \Sigma x_i \qquad s = \left(\frac{1}{N} \Sigma x_i^2 - \bar{x}^2 \right)^{\frac{1}{2}} \qquad 2)$$

Where the x_i are the data points read, N is the number of reads, $\bar{x}$ is the mean, and s is the standard deviation.

First, the means and standard deviations are calculated for the pulse data read in during the last two hours. The partial sums for this have been stored on the disk by a high priority task which

is activated after each pulse has been read in. The sums are stored separately for each of four pulse numbers (first pulse, second pulse, etc. of a multipulse sequence). Since 125 points are stored for each pulse, three disk blocks are needed for the six partial sum words per data point. These are read sequentially into the first general buffer and the resulting means and standard deviations stored in the second general buffer. The number of pulses for the A (hadron) and B (neutrino) experiments are stored
* in place of the means for the last two data points. As a special case for pulse number 0 (the first pulse), four scalars in the CAMAC crate are read, reset, and stored in place of the means for the previous four data points. These scalars are used to record the number of accelerator clock pulses, the number of accelerator pulses with beam in the Main Ring, and the number of pulses with beam hitting the neutrino target during the last two hours. These numbers have been used to study the various efficiencies which affect the picture taking rate at the bubble chamber.

When the output block has been filled, it is written on the disk in area for means and standard deviations for the first pulse. If the magnetic tape logging is active, the block is also written as the next record on the tape. After each input partial sum block has been used, the first general buffer is cleared and the current date and time put into the second through fourth words. This buffer is then written back on the disk to clear the partial sums for the next two hour period. The whole process is then repeated for the data from the second, third, and fourth pulses.

Similar operations are done next for the fast and slow data.

Six input partial sums blocks are used for each data type and the means and standard deviations are stored in separate output blocks and written into different areas on the disk. The means and standard deviations are also logged as separate records on magnetic tape. After completing operations on the slow data, the data processing task releases the three general buffers and returns to the orwait until more work is required of it.

The processing of each data point in the slow and fast data blocks is controlled by a data status byte. These bytes are stored in one block on the disk and read into a permanent block in memory at program start up time. The result of the values assigned to this data status byte are shown in Table VI. The value of 0 is usually used to turn off the read of a bad input, for example an open TC. Three is the usual value for an input which is functioning correctly. Other values can be useful when debugging and setting up the control blocks on the disk.

* The read in, analysis, and logging of data described in this Section are all automatic and require no operator intervention. The information needed by the program is stored in blocks on the disk (see Table VII) which can be set up before a bubble chamber run by one knowledgeable person. Permanent changes to these blocks to, for example, read and process a new piece of data or change the calibration constants for an existing piece of data can be made quickly while the online program is running without interrupting the data processing. Since the information is on the disk, such changes do not have to be made again when the program is restarted. These 54 blocks stored on the disk represent about

one half the 112 blocks of memory on this computer, which is the maximum amount for a PDP-11/20. Using disk storage in this way enables the program to handle four times the data points in a much more general way, as compared to the code described in the next Section. Both sections of code require about the same amount of memory.

VI. READING AND PROCESSING OF PULSE DATA

* This Section describes the handling of data which is acquired and stored for each pulse of the bubble chamber. Unlike the slowly varying data discussed in the last Section, chamber parameters vary rapidly during the pulse and time is critical in correctly reading them. Consequently, the coding which does this job, is quite different in character than the rest of the online program. Critical parts of the code have been written to be fast at the expense of requiring somewhat more memory. Input and output buffers are permanently assigned, instead of being available for general use. The code is entered by an interrupt within a few microseconds after the electrical signal which starts the bubble chamber expansion system. Once entered, the interrupt code runs on the highest priority and retains use of the central processor for the duration of the chamber pulse (about 350 ms). Because of time requirements and the specialized and varied character of the data read during the pulse, all operations have been written out in code and no tables, etc. are read from the disk during the pulse. The one block of constants used is read when the online program is started and retained in memory at all times. The code contains three blocks (3x256 words) of output buffers which can hold the data from six pulses. The three blocks are organized with an identifying word first, followed by a date word. Next, are two time words, followed by 125 data words, and this is repeated for the second pulse in the block. The writing of this pulse data from the output buffers onto the disk and magnetic tape, the addition of the pulse data to partial sums blocks (for calculation of the

two hour means and sigmas for each pulse number), and the display of the pulse data are done by high priority tasks after the pulse is over and the interrupt code has released the central processor. The interrupt code, with its buffers and the one constants block currently require 3586 words of memory; the high priority tasks specifically for handling the pulse data require another 900 words plus one general buffer.

Figure 2 shows the various steps taken by the code after the interrupt occurs on the expansion value open electrical signal. Also shown are a typical pressure curve inside the chamber vs. time and the times that beam enters the chamber and that the lights are flashed. Immediately after the interrupt occurs, eight channels of analog information are latched in sample and hold amplifiers and a direct memory access read of this data is started by the fact analog to digital converter. Usually, six of these channels receive data from dynamic pressure transducers at various locations inside the bubble chamber, between the piston rings, and under the piston. The remaining two channels measure the position of the chamber piston. The latch of the analog data occurs in a few μ seconds, the read in of eight channels requires about 200 μ seconds. Since the chamber piston doesn't start to move until a few hundredths of a second after the expansion valve open signal , these readings give chamber pressure and piston position before the pulse starts, i.e. their static values.

Next, the time since the previous interrupt, is determined using the computer's internal programmable clock, which runs at 10 KHz. If this time is greater than a preset value in the constants block

(two seconds) the pulse is assumed to be the first pulse of a possible multipulse sequence, if the time is less, then the pulse is a second, third . . . etc. pulse of the sequence. A pulse number is now assigned for this pulse, Ø means the first (or only) pulse in the multipulse sequence, 1 means the second pulse, 2 the third, etc. Separate timing constants for five different pulse numbers are stored in the constants block and the correct set is now transferred to the active timing locations. These timing constants determine the length of steps A, B, and D, the length of the early, beam, and late gates, etc. Next, the roll and frame for both the A (hadron) and B (neutrino) experiments are read as well as the precision static pressure transducer and the magnet current. The next pulse output buffer (½ block long) is selected and marked to show the three high priority tasks that it contains new data. The static values of the eight A/D channels are converted to physical units (i.e. PSIA or inches) and transferred to this output buffer. Also transferred is the other data read in so far and the timing constants described above.

The time remaining for step A, to the nearest 100 μs, is now calculated (see Figure 2) and the processor goes into a loop which keeps checking the clock until the necessary time has gone by. Every 500 μs, the state of eight binary bits is checked and the time of any changes is recorded. These eight bits monitor important events during the bubble chamber pulse and are given in Table VIII. The checking of these eight bits is done every 500 μs in steps A, B, and D; in step C it is done every 100 μs to get more precise values for beam and light flash times.

Step B is started after the time for step A has elapsed. This should be shortly before the chamber piston starts to move and the chamber pressure starts to drop. The eight A/D channels are sampled every 500 μs and then read using direct memory access by the fast A/D converter, which takes about 200 μs. After sets of data have been read into two separate eight word buffers, analysis of that data begins while the next set of data is being read into a third eight word buffer. This analysis is done during read in, both to minimize processor time at the highest priority level, and to save core by using the same three eight word buffers over and over again.

* An important piece of information about the bubble chamber is how much work is done by the expansion system on the chamber liquid during the pulse. If too little work is done, tracks will not be visible; if too much, track and parasitic bubbles will be too large and the required additional cooling of the chamber liquid will result in increased schlieren effects (local distortion of tracks and fiducials due to temperature gradients in the chamber liquid near the cameras), and may even overload the hydrogen refrigerator. The work done on the chamber liquid is:

$$W = \int_{\text{cycle}} PdV$$

which can be approximated by (see Figure 3):

$$W = \frac{1}{2} \sum_{\text{cycle}} (P_{i+1} + P_i)\ (V_{i+1} - V_i) \qquad 3)$$

Data from the two previous reads is used to add one term to the sum in equation 3, while the fast A/D is reading the current data into the third buffer. Actually, there is enough processor time between reads to calculate three different $\int PdV$ values, using three different

pairs of transducers. In addition to these calculations, the data from each transducer is checked to see if it is a new maximum or a minimum for this pulse and, if so, the value and time are recorded. The coding for these calculations, controlling the fast A/D, and checking the eight bits for timing, has been carefully written to be as fast as possible in order to finish in 50 μs. If any time remains at the end of the cycle, the processor loops on the clock until time to start the next 500 ms cycle.

Step C is started 20 ms before the time that beam is expected to be injected into the chamber. The data from the fast A/D is switched to go into an array of 81 eight word buffers and the analysis described for step B above is no longer done between the 500 ms reads. This is done so that the processor can be used to check the eight bits for timing every 100 μs and to read two scalars which are intended to record hadron beam particles entering the chamber. It has the additional feature that the most interesting part of the eight channels of transducer data, i.e. at beam time ±20 ms, is available for possible further analysis after the pulse is over. The time near to beam time is divided into early, beam, and late
* gates. Under typical operating conditions, the bubble chamber is track sensitive to particles arriving from about 10 ms before beam time, to within a millisecond of the time the lights are flashed. Early tracks will have low bubble density and a large bubble size; late tracks will have small bubble size. Out of time tracks are hard to analyze, especially for cross section measurements. The hard wired bubble chamber gating can be set to inhibit taking any picture with early or late hadron beam particles; recording their presence with the computer and taking the picture provides

physicists with the option of excluding those pictures from parts of their analysis at a later stage. The signal from a coincidence of scintillation counters in the beam is fanned out and counted by two 100 MHz scalars in the CAMAC crate. One of these is read and cleared at the start of the early gate to record counts occurring since the last chamber pulse while the chamber was not sensitive. It is read and cleared at the start of the beam gate for early counts, read without clearing in the center of the beam gate for counts in the first half of the beam gate, read and cleared at the end of the beam gate for the total number of counts in the beam gate, and finally read and cleared at the end of the late gate for late counts. The second scalar is used for more precise timing information. It is cleared at the start of the early gate, then read and cleared every 1 ms during the early gate. All the data from the early gate is packed into one computer word with the appropriate bit set if there were any counts during that time slot, and the bit cleared if there were none. During the first half of the beam gate, another computer word is used and the time slots are reduced to 100 μs. The last half of the beam gate uses another word and 100 μs intervals and finally a fourth word and 1 ms interval is used for the late gate. This process is duplicated for a second coincidence in the other two channels of the quad 100 MHz scalar in the camac crate. The CAMAC input is done by direct commands to the CAMAC interface, rather than using the BISON library program KSOO11[11], in order to save time.

Step D is started at beam time plus 20 ms. The read in and analysis of data is done exactly as during step B, which is described above. Step D should continue until the chamber piston has

returned to its initial position.

Now that all the data about the pulse has been read in, Step E is started. First, the A/D data stored in the 81 eight word buffers during step C is analyzed in the same fashion as was done in real time during steps B and D. Next, the scalar, timing, $\int PdV$, and extreme pressure and stroke values, together with the times these extreme values occurred are converted to the proper form and stored in the output buffer. This includes converting extreme pressure and stroke values to physical units using calibration constants stored in the constants block and converting the $\int PdV$'s to joules using the calibration constants for the two transducers used for the integration. Times are recorded in 100 μs intervals after the start expansion signal. If an event didn't occur, the data was too big, etc.; the appropriate error code is put into the output buffer in place of the data.

The actual time of the beam timing signal, as recorded from one of the eight timing bits, is compared with the time expected as defined by the constants for this pulse number stored in the constants block. If necessary, a change is made to the constant controlling the length of step A. Likewise, the difference between the flash and beam times is compared and the length of the late gate adjusted if required. Other timing changes are not normally required, if necessary, they can be made manually in the constants block, via commands entered on the CRT keyboard.

The time of the minimum value from one pressure transducer is selected to define when the minimum pressure occurred. The time intervals between beam and pressure minimum and between beam and flash are then calculated and stored in the output buffer. One

pressure transducer and one stroke transducer are selected and the difference between the appropriate extreme value and the static value used to calculate the pressure drop and stroke, which are stored in the output buffer. The selection of transducers is determined by entries in the constants block. The beam to pressure minimum time, pressure drop, and stroke, as well as one of the three $\int PdV$ values is sent to a numeric display in the rack above the expansion system controls. The data is sent to the display serially, using eight bits of a digital output module in the CAMAC crate. The last three data are converted to analog voltages and available for recording on a strip chart in the Control Room. The * pressure drop has turned out to be an accurate gauge of chamber sensitivity and is used frequently by the operating crew to maintain stable operating conditions during physics running and to quickly reestablish proper operating conditions after periods of downtime. In the latter case, 15 to 30 minutes of valuable beam time is saved because a test strip is no longer required before starting physics pictures.

Event variables are then set for three high priority tasks which are responsible for further operations on the data in the pulse output buffers. Finally, the CAMAC device handler is notified that it has been interrupted and that it will have to repeat any routine I/O operation that it may have been handling. The interrupt operation is then ended and the processor released to continue its normal task processing under the BSX system. The three high priority tasks are responsible for: 1) writing the pulse data on magnetic tape; 2) writing it on disk, updating the partial sums (used to calculate means and standard deviations for each pulse number every

two hours) on the disk, and updating partial sums stored in memory for a few key parameters (used to calculate means every few minutes); 3) updating a listing of pulse parameters on the memory scope (if requested) and/or a page listing of a subset of up to 15 pulse parameters on the CRT display (if requested). Since these high priority tasks operate on the data in the six pulse output buffers, they need not finish their work before the next pulse interrupt occurs. Up to five pulses interrupts from one accelerator beam spill are stored in the pulse output buffers and the three tasks will complete their work on them before the next accelerator beam spill without losing any data.

* A special effort has been made not to require any action by the bubble chamber operators in order for the computer to correctly read and store the data from the pulse. All the control information needed by the program is stored in the constants block, the latest version of which is read into memory from the disk automatically when the online program is started up. This block is typically set-up at the start of a run by one knowledgeable person and requires a little modification during the run. The timing signals used (start expansion and beam times) are the same ones that the operator must set correctly for proper bubble chamber operation, whether or not the computer is running. The operator has no additional timing settings to make for proper computer operation. The data logging is automatic and available for future display, subject of course to the space limitations of the disk. Updating of the numeric display in the expansion control rack is also automatic. The display of more complete information is optionally available on demand.

VII. DATA STORAGE

* In addition to small amounts of data storage in memory, the online program uses the disk to store large amounts of data which can be recalled quickly and the magnetic tape to store data permanently for later offline analysis. One large contiguous file (BCDATA) of 4096 (256 word) blocks on the disk is allocated for use by the online program. As shown in Table IX, this file is organized into sixteen 256 block groups. Fourteen of these groups are used to store various types of data about the bubble chamber. Each group is used as a circular buffer with the most recent data being written over the oldest data of that type. One group is used to store control tables, calibration constants, temporary results, list specifications and other information needed by the online program, see Table X. Storage has been allocated so that all of the most recent data is available for a few hours, a less frequent sampling of the data is retained for a couple of days, and data means and standard deviations are available for three weeks. The 16th group of 256 blocks is available for future use. The remaining 704 blocks on the disk cartridge hold the DOS monitor, DOS system programs PIP and VERIFY[3], needed to transfer files between peripherals and to verify the disk file structure, and the load modules for up to three versions of the online program. Data is logged on the magnetic tape just after it has been read in by the online program in one long file. The online program automatically writes a standard PDP-11 file label at the start of each new blank tape. The file name includes a sequential serial number which is automatically incremented for each new tape. The creation date is also included in the file label. Next, the first 256 blocks (i.e. the first

group) on the disk, which contain the control tables, etc. are written on the new tape. This is done to provide a complete description of the data on the tape for the (future) offline analysis programs as well as to make a backup list of these control blocks which can be used to restore the file on the disk if necessary. If the program is restarted with a partially filled magnetic tape, operator commands are available to properly position the tape before data logging on it is resumed.

All blocks in the file on the disk and records on the magnetic tape (except the file label record) contain 256 words and have the same identifying information in the first four words. Word 0 contains a four bit code in bits 15-12 indicating what type of information is stored in the block, see Table XI. The 12 low order bits contain the block number, relative to the start of the 4096 block BCDATA file, on the disk where the information is stored. This is included to further specify the contents of the block and to make restoration of the disk file BCDATA from the magnetic tape easier, should this feature be required in the future. Word 1 contains the date and words 2 and 3 the time, in PDP-11 format[8], when the data was created or the control list last modified. Usually, the remaining words contain the data or control list, see Table XI for exceptions.

Data blocks in all the data groups in Table IX, except the last ones for the values of pulse data, are stored within the group at a position determined by the clock task (see section X) based on the date and time that the read in of each data block began. If the online program was not running for some period of time, no data

will be read and no data will be stored in the relevant locations in the various groups. These locations will still contain earlier data. Before any data from these groups is used for a display, the date and time words (words 1, 2, and 3 in the data block) are checked to be sure that the data was indeed read in at the expected time. If this check fails, the error code for no data is output instead of the data, or if no data at all is available for a list, a no data error message is output on the CRT screen.

The data storage for the values of pulse data (last entry in Table IX) is sequentially with the pulses as they occur and not time based. Therefore, no date-time checks, as described above, are made before the data is displayed. A binary search of the 512 blocks is done during program start up to find the most recent data, and storage of the next two pulses will be in the next block. This means that the most recent 1024 pulses stored by the online program are available for display, even if the program has just been restarted. Blocks 3584 to 4095 are used as a circular buffer; if the previous two pulses were stored in block 4095, the next two pulses are stored in block 3584.

During the start up phase of the online program, standard DOS program requests[3] are used to find the disk address and length of the **BCDATA file.** These are transmitted to the subroutine which handles all disk input and output for the online program tasks. When a task needs a disk I/O operation, it calls this subroutine specifying the relative block number in the BCDATA file, the start address of the (256 word) block in memory to be used for the transfer, and either read or write as arguments using the DOS Fortran call convention[3]. The front end of this disk I/O sub-

routine is re-entrant and, if the subroutine is already busy for another task, will cause the task issuing the new request to go into a BSX wait until the subroutine is free. When the disk I/O subroutine is free, BSX will give control of the CPU to the highest priority task with a pending request. BSX can handle multiple I/O requests to the same device directly, but this would require more code (i.e. memory) in each task and would not honor the pending requests in order of their task priority. The disk I/O subroutine stores the relative disk block number in bits 11-0 of the first word in the block (writes only), checks that the requested block is within the BCDATA file and issues a fatal error message if it is not, calculates the actual disk address, and uses the BSX QTRAN directive[6] to start the transfer. The requesting task then waits until the transfer is complete (notification is via a BSX event variable) and then control is returned. The disk I/O subroutine requires 67 words of memory.

When magnetic tape logging is active, program requests to log data are handled by a high priority task. This task is activated either by a pulse interrupt handler, or by a subroutine similar to the disk I/O subroutine described above. When activated, it searches the pulse output buffer control words for pulse buffers which have not been written on tape yet, writes those buffers on tape, and then marks them empty (in so far as the tape writing task is concerned). If a general request to write data on tape is pending, it will be accomplished now by the tape task and the tape output subroutine notified via an event variable. The tape writing subroutine and that part of the tape task in the main program require 233 words of memory.

* Tape positioning and tape label writing, etc., requires 1101

words in the first overlay. These functions are only needed occasionally, so putting the code in an overlay results in a net saving of memory. This code is rather longer than is absolutely necessary to satisfy two desires. First was to make it possible to resume data logging on a tape at the point where it was interrupted by power failure, program bomb-out, tape unit cleaning operation, etc., so that each tape would be full, instead of having a larger number of partially filled tapes. Fewer tapes means less cost, less volume required to store them, and less time and operator effort required to change tapes. The second desire was to make the tape handling commands as simple as possible while still providing sequential, DOS tape labels and reducing the risk of data loss through operator error.

When the online program is started up, or when the operator enters a command to start logging data on the magnetic tape (after this feature has been inactive), the mag tape status register[2] is tested and error messages written on the CRT terminal if the tape unit is not ready or there is no write ring in the tape. Once any such problems are resolved, the register is tested to see if the tape is at the load point or not. If the tape is at the load point (beginning), the tape unit is ordered to skip forward one record. If the tape is blank (new), this would normally result in a runaway tape unit, which would skip forward through the entire tape (about 10 minutes). To prevent this, the magnetic tape byte record counter is tested every 1.0 second and if it has not changed, a power clear command is issued to stop the tape. After one second, a fake "operation complete" interrupt is sent to the driver routine to reset it and the tape is rewound. Since the tape has been

verified to be a new blank tape, a file label record is written on it containing the current date and a sequential tape number. This sequence number was stored in the general constants block which was read in from disk block 0 earlier in the program start-up phase. The number in the constants block is now incremented and the updated constants block written back on the disk to be ready for the next new tape. Next, a general buffer is reserved and the first block in the BCDATA file is read from the disk into the buffer and then written out on the tape. This procedure continues until the first 256 blocks on the disk have been written on the tape, then the general buffer is released. Data logging on the new tape now begins automatically.

If the original skip operation ended normally, the tape is rewound to the load point and the first record read and summarized by a message on the CRT screen. In the usual case that the record is a DOS file label, the CRT message includes the file name and creation date. The operator then has three options. If he changes the tape, the program notes the not ready condition of the tape unit and automatically starts over again with the procedure given in the previous paragraph. If he gives the command to overwrite the tape label and data, the program treats the tape as if it were originally blank and proceeds as outlined above. If he wishes to move the tape to the end of the recorded data, he uses the commands given below.

If the tape was not at the load point when the online program was started-up or tape logging requested, the tape is backspaced one record using direct commands to the hardware registers. This special backspace is used to avoid fatal errors from the DOS driver

if the tape was positioned in the middle of the record. If the backspace reaches the beginning of the tape marker, the program then treats the tape as if it was originally at the beginning. Otherwise, the record is read from the tape and a summary message written on the CRT screen. In the usual case that it was a data record written by the online program, the message includes the ID, date, and time information from the first four words of the record, see Table XI.

Once the summary message of the record has been written on the CRT screen, the operator has several commands available to position the tape and indicate where logging of data is to resume. The tape unit can be moved forward or backward a specified number of records. On forward moves, the program makes the checks outlined above for blank tape and stops on blank tape, on end of file (EOF) mark, at the end of tape or when the specified record count is satisfied. After either move command, the previous record on the tape is read, summarized on the CRT screen, and the tape positioned just after that record. When the operator has moved the tape to the desired position, another command starts the logging of data on the magnetic tape. Unless canceled by the operator, the first 256 blocks from the disk are written on the tape before data logging is resumed.

The default option for the skip forward command is for a very large number of records (more than the tape holds), so this one command alone is sufficient in most cases to position the tape at the end of the data previously written on it. If data was being logged over old data on a tape and no EOF mark was put on the tape at the end of the new data, the operator must find this point by using

the tape positioning commands and studying the information in the summary messages.

Once tape logging has begun, it normally continues until the end of the tape marker is reached. The program then backspaces the tape, writes three EOF marks, rewinds the tape, and writes a message on the CRT scre requesting a new tape. When the new tape is loaded, the program will automatically start the procedure described in this Section. Commands are also available to manually start the end of tape procedure (to permit tape unit cleaning), specify no tape logging (if the tape unit is down or for program debugging), or to restore the disk control tables from the tape.

VIII. DATA DISPLAY

* Most data displayed by the bubble chamber online program is optional and requested by operator command. Since the code to make these displays is only needed when these requests are made, most of it is overlayed, in fact it makes up the bulk of the three overlay segments. The addition of more optional displays and special optional analysis of data would be relatively easy, because additional overlay segments can be added with little or no increase in memory requirements. Although the most recent data is kept in memory, most of the data available for displays is stored on the disk (917,504 words). Rates are not stored on the disk, but are calculated each time they are needed for a display.

Much of the data display requires the output of alphanumeric characters on one of the four line output devices: the line printer, memory scope, CRT terminal screen, and Decwriter. This data is formatted using the Bison routine FMTPUT[10] which I have modified to insert a decimal point in the data if desired and print out an error code, see Table XII, if the data value is in the range 100000_8 to 100007_8 instead of the actual value. By specifying the previously unused code 5 in bits 7-5 of the format input to FMTPUT, a signed decimal integer conversion with these features is done. The number in bits 14-13 of the format specifies the number of digits to the right of the decimal point. The resulting displays show the data with decimal points in the usual, quickly understood format, while still storing the data as single precision integers, which is virtually required by the size and hardware of the PDP-11. The modified FMTPUT requires 471 words of memory. All requests by tasks to output a line of data are handled through a line output

control subroutine for several reasons. The subroutine reserves the line output device for that task for a preset time interval and puts tasks requesting the same device in the BSX wait state until the time interval is over. This prevents mixed lines of output for two different lists on one page, and also leaves pages of output on the memory scope for at least a minimum time so they can be examined. The subroutine inserts form feeds and title, date and time information if requested. Finally, since FMTPUT is not re-entrant, the subroutine will allow only one task at a time to use it. The subroutine uses the BSX QTRAN directive[6] to output the line after it has been formatted. The line output control subroutine requires 679 words of memory, which include line output buffers for the four devices.

One principal means of displaying data is by a list of many different data points, all read in at (approximately) the same time. These lists can be output either on the memory scope for temporary use, or on the line printer for a permanent copy. The list can include rate information, based on the difference between data read some interval (15, 60, or 120 minutes) earlier and the values at the selected time, and expressed in units per hour. These lists are displayed on a relatively simple command typed on the control CRT terminal keybroad by the operator. The command consists of one or two letters specifying output on the memory scope or line printer, a list number, a letter specifying the type of data desired (C = latest read, A, Q, or M; see Table IX), and the desired time expressed in days ago, hour of day, and minute of hour. In the case of a request for pulse data, type A, the number of pulses ago is specified instead of the time.

The command is converted by the Bison program CICONV[9], the first overlay read into memory, and the command decoded and stored by code in the overlay.

An event variable is set for the low priority lists task and the command task returns to wait for another command. The list routines occupy 591 words in the main segment and 1137 words in the first overlay. The bulk of the part in the main segment finds and reads the requested data blocks from the disk and calculates the rates. It was put in the main segment because it is also used by the few variable summary code in overlay 2. The list number in the command specifies the disk location of the first list specifications block. A general buffer is reserved and the first list specifications block is read into it. The last eight words of this block contain data block codes which specify the data blocks needed for the list. These can be slow, fast, or pulse data, any specific block in the BCDATA file, or the data in the edit buffers (see Section IX). In the first case values, standard deviations, or (15, 60, or 120 minute) rates are also specified in the data block code word. The required additional number of general buffers is now reserved and the desired data is read into the general buffers. For requests for bubble chamber data, the data type, date, and time specified in the list command as well as the data block code word are needed to specify what data is needed. The date and time of these data blocks are checked and if they do not match the expected values, flags are set to output no data error codes in place of the data. Any desired rates are now calculated.

The code words starting in word 5 of the list specifications

block are now scanned, the lines of the list constructed, and the list output line by line. Before the first line is written, the line output control subroutine is requested to send a form feed and title containing the current date and time to the device (which was selected in the operator command). The strings of list specification code words consist of a control word first followed by the required number of words specifying data location or alphanumeric characters to be output as titles. The control word is very similar to the format word needed by FMTPUT[10] to format the desired data, but some bit combinations which are unused by FMTPUT are used by the list routine to order special operations. The number and use of the words following the control word are determined by the repetition count and format code fields of the control word, which are the same as the FMTPUT format word definitions. Data location words specify one of the eight possible data blocks (defined by words 248-255 of the first list specifications block) to use and the offset of the desired data within that block. A zero control word ends the line on the list, and a second zero control word indicates the end of the list specification block. If word 4 of the specification block is zero, the list is complete; if the word is nonzero, that block is read in and used for continued list specifications. If the list was on the line printer, a two second timer is started and if no further lists to the line printer are begun in that time, the list task is re-entered and it spaces the paper forward 25 lines to move the printed text out of the toner tray.

As pointed out in Section IV, the main motivation for using this method to display lists was to put the list specifications on

the disk, thus relieving most of the memory requirements for this job. Presently, there are 15 lists using 28 disk blocks for their specifications, but this **can easily be increased without** requiring any more memory. Another benefit is that new lists can be added and old lists modified while the online program is running, see Section IX. The data from the online program is so useful in operating the bubble chamber that the operators dislike any interruption, even for program improvements, so this feature is of real importance.

Once the desired list at the selected time has been printed, the operator can use the advance command to print out the same list using data from the next (or previous) read interval. Optional arguments of the command allow skipping read intervals and specifying how many times the command is to be automatically repeated.

Figures 4 and 5 show examples of bubble chamber data output with this list program. Figures 6, 7, and 8, also made with this program, show the data that is currently read into the slow, fast, and pulse data blocks. For the first two, the online program uses the same two or three letter plus number code to name indicators that are in general use at the 15' bubble chamber.

The 15 lists currently implemented include seven which output data about the bubble chamber and its support systems and eight to give information about the set-up of the online program. Of the latter, three are used to give the read lists (Figures 6-8), four give a list of the limit checks which are presently active or acknowledged, and one gives a dump of the edit buffer.

A special optional provision is made to output the list of gold chromel thermocouples on the memory scope each time the slow data

read in is completed (16 times per hour). This is used primarily during cooldown when this information is vital for the operator to correctly control the cooling rate of the glass bubble chamber windows. A similar provision is available to automatically list data taken every chamber pulse on the memory scope, but this is less useful. When the chamber is being double pulsed, there is insufficient time to print out all the data on the memory scope.

From the data taken every pulse, the operator can select up to 17 data points to be output as a page (similar to pages in the beam line MAC system) on the CRT screen. Fourteen pages are possible, and this page display is activated by a one letter command followed by a page number. This command calls the first overlay into memory which then reserves a general buffer and reads the block of page definitions. The data point numbers (word offsets in the pulse data 1/2 blocks) for the requested page are taken from the definition block and saved in memory locations within the main segment. The pulse ID blocks are then read sequentially into the general buffer and the output format (which contains the decimal point location, i.e., the scale factor) for the desired data points is also saved in the main segment. The 14 character title for the data point is taken from the ID block and written on the proper line on the CRT screen. Since this title is then stored in the CRT terminal memory, there is no need to save it in the computer memory. When all the ID blocks have been processed, the general buffer is released, a flag set to show that the page display is active, and the overlay is released.

After every pulse of the bubble chamber, the pulse data display task is activated and will update the CRT page display. Using the

data point numbers and formats saved when the page was requested, this task deletes the seven characters on each line just after the title and then writes the desired data at the end of the line. After five chamber pulses, each line contains the title of the data point displayed on that line and the five most recent values of the data point, earliest one on the left and latest one on the right. This gives the operator information on the pulse to pulse variation of the data and the CRT display is fast enough to output even multipulse data between accelerator cycles. As a special option, the hadron beam hits/misses vs. time can be displayed on the 18th line of the page; time increases from left to right and X's appear in time slots with at least one hit, see Section VI. The line after the normal data point displays is deleted before the line is written, so that several of these timing lines can be displayed with the one from the latest pulse at the bottom. Figure 9 is an example of this display, using simulated data when the bubble chamber was not running.

As described in Section VI, four key chamber expansion parameters (stroke, $\int PV$, pressure drop, and beam to pressure minimum time) are sent to a 16 digit display in the rack above the expansion system controls every pulse. The first three of these parameters are converted to analog voltages and are available for recording on strip charts.

In addition to the lists, which display a large number of data points at one time, it is possible to display a summary of a few variables (up to 7) at many successive times. Frequently, knowledge of the time variation of a few variables is important in understanding some aspect of the bubble chamber operation. To obtain such a

summary the operator enters a command just like the list command described above with a special list number. The command is decoded by the same code in overlay 1 as was used for a list command. The tiem specified is that for the end of the summary and the default time is now. Since this would be the usual request, no time need be entered in most cases.

After the command is decoded, overlay 2 is called and the code there reads in a block of information from the disk which contains the list of variables which were used for the last summary. The seven old variable definitions are output one at a time on the CRT screen as a command to define a variable for the summary list with arguments specifying the old definition (see Figure 10). This allows the operator to change the definition, but if no change is needed, he need only press the return key. Specifying the variable completely requires one letter to indicate the slow, fast or pulse data blocks, the data point number (see Figures 6, 7 and 8) and a three character code for value, standard deviation, or (15, 60 or 120 minute) rates. Once the variables are specified, a line is written on the screen with the number of entries and the interval in minutes used before. Changes to these parameters can be made before the return key is pressed. If the interval is specified as 0 minutes, the program will compute the minimum interval for which data is stored, using the first variable definition and the data type specified in the list command. Next, an event variable is set for the lists task and the command task returns to waiting for the next command.

The remaining code is also in overlay 2 and runs on the lowest priority lists task in order not to interfere with the data read in

and logging functions of the online program. Abstracting the data from the disk blocks may take up to one minute if the maximum number of 256 entries is requested and data is needed from blocks in several different groups on the disk. First, the specified variable list is scanned and the required set of code words, identical to those stored in the last eight words of the list specification block, needed to obtain the complete list of variables is constructed. The output format and the information for a 20 character title for each variable is read from the various ID blocks on the disk and stored in the same block as the variable definitions. Using the input arguments, the date and time for the start of the list and the interval between desired data is calculated. The subroutine in the main segment is then called to read the necessary data and calculate any required rates. This subroutine is the same one used for the list display and is described earlier in this Section. Since the subroutine may require up to three general buffers for each of the slow, fast, and pulse data types (if all were in the requested variable list), sequential calls are made for each data type and the desired data is abstracted after each call. This abstracted data as well as the date and time words from the block containing the first requested variable are stored in a general buffer. If the block of data containing the first variable was never read, zeros are stored in place of the date and time words to show that no data is available. This procedure is repeated for subsequent times as specified in the operator commands. When the general buffer is filled, it is written on the disk in the area for this type of data (blocks 131-142). The block of constants, which

now contains the variable definitions, titles, and formats for the new data, is also written back on the disk.

Finally, the data for the summary is read back sequentially into a general buffer and the list output on the requested device. This step is skipped if the no list option was selected in the original command. In any case, the data summary on the disk has been updated and is now available for the graphical displays described below. 1244 words in the second overlay are required to make this few variable summary. Figure 11 is an example of such a summary. The actual commands used to make it are shown in Figure 10.

Once the few variable summary list has been made, the data on the disk is available for plotting. Plots on the memory scope are made using the Bison subroutine PLOTA[12]. The operator enters a simple one letter command, followed by a one letter option and the variable number(s) which are defined when the few variable summary list was made. If the option or variables are not specified, the previous definition is used. Options are for one large line graph of a variable vs. time, with or without symbols plotted on the points and with optional error bars taken from a second variable. The latter option only **makes sense, of course, if the plotted** variable is a mean and the error bars are taken from the corresponding standard deviation. Also, four small line graphs of different variables can be put on the memory scope at the same time, either with or without symbols plotted on the points. A scatter plot of one variable vs. another can also be made. In this case, the coordinate pairs come from data read in at (essentially) the same time. Titles and tic values are also put on the graph. The hard-

ware and subroutine HCOPY[4] supplied by the computing department can be used to copy the graph from the memory scope onto the line printer, but resolution suffers somewhat in this this transfer. Figures 12-15 were made by the online program using this feature. The code to plot the graphs on the memory scope and make the hard copy is all in the second overlay and requires the remaining 2481 words in that overlay.

When the operator enters the command to plot on the memory scope, the second overlay is read into memory and the block containing variable definitions, the previous graph option and variables, etc., is then read from the disk into memory. The graph options and variables are updated from the command, if necessary, and the required three or four general buffers reserved. The data blocks written on the disk when the few variable summary was made are then read sequentially into one of these buffers and the required data for the graph is put into the remaining buffers in the proper format for PLOTA. One buffer contains the X coordinate, another the Y coordinate, and the third (if required) contains the corresponding error. The memory scope is then reserved, erased, and the X axis title written on the scope using the line output control subroutine described earlier in this Section. The remaining arguments are then set up, and PLOTA called to output the line graph, Y axis title, tic marks and labels. If symbols are requested, the arguments are modified and PLOTA called again to put them on. If the option for four small graphs was selected, the above steps have output only the first of these. The read of the few variable summary disk blocks and general buffer fill is repeated for the second graph, arguments modified and PLOTA called again (twice for

symbols on the points) to plot the second graph. This is repeated for the third and fourth graphs. Finally, the data block with the updated graph option and variable numbers is written back on the disk and the general buffers released.

On a command to make a hard copy, overlay 2 is read into memory and an event variable for the list task is set. The lists task then calls HCOPY to copy the memory scope onto the line printer. Since HCOPY was written with wait loops rather than interrupts to signal the end of each I/O operation, it is necessary that it runs on the lowest priority list task to permit the read in and data logging functions of the online program to continue without delay. It takes about 20 seconds to make a hard copy.

It is also possible to make graphs of one variable vs. time directly on the line printer using the Bison program PLOTB[5]. The resolution of such graphs is considerably better than those made with the hard copy feature. The online program handles such requests in a way very similar to that used to make graphs on the memory scope, with the following exceptions. The code is in overlay 3 and the entire overlay is needed to make the line printer graphs. The option to make four small graphs is not available or necessary; the operation can enter four commands if he requires a permanent record of four graphs. PLOTB was not written to make scatter plots, so this option is not available. This job is switched to the lowest priority task after the command has been decoded. This is necessary because the plotter driver or hardware supplied by the computing department does not currently work on PDP-11/20's, so I have written simple instructions, similar to those used in HCOPY, to transfer the data to the line plotter registers directly. The wait loops

tie up the task for the 20 seconds or more it takes to output the graph. Since the other tasks all have higher priority, this doesn't delay the read in and logging features of the online program. Another problem with PLOTB is that it requires a full character line for each X axis value, so a graph of all the data available (256 time values) requires five pages. Figures 16 and 17 were made directly on the line printer by PLOTB and contain the same data as Figures 12 and 13. Note that the actual commands needed to make Figures 11, 12, 14, and 16 are shown in Figure 10.

IX. DISK BLOCK EDITING

* Since the bubble chamber online program makes extensive use of control tables, constants tables, and list specification blocks on the disk, some feature is necessary to enter and modify the information in these blocks. This could be theoretically done with a separate program, but there is a distinct advantage in being able to do disk block editing with the online program. The BSX task structure permits the disk block editing to be done by the online program with no interference to the read in and data logging functions of the program. A separate program would mean that these functions would not be done while the editing program was being run to edit disk blocks. The data from the online program is so useful in operating the bubble chamber that any interruption, even for program improvements, should be avoided if possible, so being able to edit control tables and list specification blocks online is of real value. More important still is that changing limit check values, which means editing blocks on the disk, is frequently done by the operators and, therefore, should be as simple and quick as possible. A separate program would take longer and require more commands to be entered, compared to the present online editing.

All the code for the editing is in the first overlay, requiring 1573 words. For direct editing, the programmer first enters a command to assign one or two general buffers for editing. Two buffers are used as separate input and output buffers. In the editing process, selected data is taken from the input buffer, displayed on the CRT screen where it can be modified if desired, and then written into the output buffer. For most editing operations, only one buffer is needed and this is used as both the input and output buffer.

These buffers must remain assigned for editing until all operations are completed, but the editing programs need not remain in memory during this period. Thus, a graph or a few variable summary (which requires other overlays) could be made without spoiling the editing process. Any editing command will recall the first overlay to get the code required to complete the requested action. The secondary editing commands, described later in this Section, do use the same code as the primary editing command, so that their use would destroy any other editing job in progress. Another command exi to assign any location in memory as the edit buffer. This is used as an aid in debugging the program online. Since the editing code was put in the first overlay to save memory, only code in the main and first overlay segments may be examined and changed with the editing commands. When the programmer finishes his editing operations, another command will release the general buffers assigned above.

Commands, specifying any relative block number in the BCDATA file on the disk as an argument, will read or write the selected block of data into either buffer. Before a write is done, the current date and time are entered in words 1-3 of the block to show when it was last modified. If tape logging is active, the block is also written as the next record on the tape. This is intended to be used by the (future) offline analysis programs for an up-to-date list of the data being logged in each block on the tape.

Another set of commands is available to take data from the input buffer, display it on the CRT screen in the specified format where it can be changed by the line edit features of the

modified KBIOHT sub-program (see Section X), and write the modified data into the output buffer. In the case of octal or decimal format, five words at a time are displayed on the screen. The word offset of the first piece of data appears before the data. The line is actually a command (written by the computer to itself) to change the data in the output buffer, starting at the word offset given in the first argument, to the values given in the last five arguments. When the return key is pressed, the Bison routine CICONV[9] will be called to convert the command, so the full power of CICONV is available to specify the format of the arguments. Since the word offset where the changes are to be entered appears as an argument, it can be changed before the return key is pressed. This provides the means to easily shift data around in the buffer. The original command allows the programmer to enter the word offsets of the first and last words he wishes to change. If these specify more than five words, sequential lines of five words at a time will appear on the CRT screen until the requested last word has appeared. For long (<45) strings of ASCII (alphanumeric) characters, conversion is unnecessary and CICONV does not accept such strings, so the program uses these directly without calling CICONV. The command for an edit of ASCII characters must specify the word offset of the first character, the number of characters and optionally the number of times to repeat the command and the number of words to skip after the end of the previous string before starting the next one. Another command can be used to change one byte of data in the edit buffer.

* One of the lists available (Section VIII) will dump the contents of the output buffer, in both octal and ASCII, on either the memory

scope or the line printer. The editing commands described above are very general and powerful, but require detailed knowledge of the program and disk block organization to use correctly. Therefore, a set of secondary edit commands have been written for general operator use to do a restricted set of editing tasks on the disk blocks. These commands call the primary edit commands, described above, when necessary. The arguments are easy to specify and require no detailed knowledge of the program or disk block organization.

Currently, there are five secondary edit commands. Four of these are used to modify the control and constants tables on the disk for one data point in either the slow or the fast data block. To specify which data point, the first two arguments after the command are one letter to specify slow or fast data block and the data point number, see Figures 6 and 7. The command to adjust the calibration constants used in equation 1 for that data point requires either two or four additional numbers as arguments. The first of each pair of numbers is the correct output for a given signal and the second is the current output for that signal. If only one pair of numbers is specified, only the zero (a in equation 1) will be adjusted, if both pairs of numbers are given, both zero and gain (a, b, and s in equation 1) will be modified. The other three secondary editing commands concerning a data point require no additional arguments. Instead, they will output on the CRT screen current values from the control or constants tables on the disk, accept changes to that information, and make those changes on the disk when the return key is pressed. One command changes the data status byte, see Table VI and Section V; the second modifies the four limit checks for the specified data point, and the third changes the read in control list

word, format, name, number, units, and calibration constants for the data point. This third command is not intended for general operator use, but has been included to save the programmer's time and reduce the possibility of an error when the disk tables are modified to cause the online program to read in a new piece of data from an existing device. The fifth secondary editing command will change the variables displayed on the specified page for the CRT screen page display of data from the bubble chamber pulse. The page number to be changed is specified as the argument after the command. The data point numbers in the pulse buffer (see Figure 8) being currently displayed are written on the CRT screen, changes accepted, and the updated page definition block written back on the disk.

All these secondary edit commands first call the code used by the primary edit command to reserve one general buffer. The arguments are used to determine which disk block is needed first and this block is read into the reserved buffer. The desired words, in the proper format, are written on the CRT screen for possible changes. After any changes are made and the return key pressed, the new information is put in place of the old in the edit buffer. In the case of the calibration command, this step is unnecessary; the arguments of the command are used to calculate the new constants which are written over the old ones in the edit buffer. If required, more lines are output to the CRT screen, etc., until all the necessary changes have been made in the edit buffer. The updated block, with the current date and time in words 1-3, is then written back on the disk using the same code as the primary edit command. In the case of the command to set up the read in of a new piece of

data, three disk blocks must be modified, so the required steps are repeated until all blocks have been processed. Finally, the edit buffer is released, the first overlay is released, and the command task returns to waiting for the next command.

X. PROGRAM DETAILS

A. Program Start Up

Several jobs need to be done when the online program is started up, and the code required to do them is not needed at any other time. Such code has all been put in one area in memory and this area is used as a general buffer after the start up phase is complete. The start up phase also uses some code in overlay 1, so this overlay is called into memory at the start of the phase. Next, the interrupt vectors are set for the pulse, scanivalve, and DVM interrupts. The CAMAC crate controller is initialized according to the procedure suggested in reference 11. The DOS .INIT, .LOOK, and .RLSE programmed requests[3] are used to find the disk address and length of the long data file, BCDATA, used by the online program. This information is stored in the disk I/O subroutine and used there before each disk data transfer to insure that the request is within BCDATA. Illegal requests result in a fatal error message, so that the online program cannot destroy other files on the online data disk cartridge or even on some other cartridge left in the disk drive by mistake. The .LOOK request requires an additional 512 words of monitor buffers which are released as soon as the request is over. Since most general buffers have not been assigned yet, these 512 words of memory will also be available for use as general buffers.

The console switches are checked and, unless a special code has been set in the seitches, the memory used by ODT will be used for general buffers. Two general buffers have already been assigned by one of the program source files. The space between the start of BXSCAN (or ODT if it is required) and the top of the monitor buffers

is now assigned as general buffers, the starting address of each 256 word buffer is stored in the buffer control subroutine, the flags for those buffers are cleared to show that they are available, and the count of available buffers is increased by the number of buffers assigned. Fourteen general buffers are assigned here, or less if the available free space in memory will not allow that many. After buffer assignment, the DOS and BSX tables are modified to hold the new program low address and stack address. A message is then written on both the CRT screen and the Decwriter giving the current date and time, total number of general buffers, and the location and length of the BCDATA file on the disk.

Next, the latest general constants block (mostly used for the pulse interrupt handler, Section VI) is read from block Ø of BCDATA and the latest status byte block (for slow and fast data, Section V) is read from block 10. The 32 character title from the general constants block and the date and time it was last modified is written on the CRT screen. The subaddress to be displayed on the DVM and both thermocouple systems, when they are not reading data are also stored in the general constants block. Event variables are now set which cause the respective drivers to output these subaddresses. Finally, a binary search of the 512 block group on the disk holding data from the last 1024 chamber pulses is done to find the most recent block written there. The data buffer containing the next two pulses will be stored in the next block in the group on the disk and the display routines will use this most recent block number when displays of all pulse data are requested. This insures that the last 1024 pulses are always available for display, even immediately after program start up. Finally, now

that the general constants block has been read in, the mag tape logging task is requested.

Once the above jobs have been completed, the memory area containing the code, formats, etc., used in the start up phase is released for use as a general buffer. Each task contains some initialization code, which is usually just to put the task in the wait state on the proper event variables. See reference 6 concerning the tasks for the standard I/O devices and below for the clock task which is more involved.

B. Clock task

On program start up, the clock task calls a subroutine which sets up several peripheral devices for the online program. The Bison program CKINIT[8] is called to change the mode for the programmable clock, direct the clock interrupts to the Bison interrupt handler, and cause these interrupts to occur every ½ second. I have modified CKINIT so that the clock runs at 10 KHz (instead of 100 KHz) to simplify the coding in the pulse interrupt handler, see Section VI. The Decwriter keyboard, which is not used by the online program, is disabled and the CRT terminal keyboard interrupt is enabled. The fast A/D converter is initialized, the scanivalve interrupt enabled to the Bison interrupt and gate unit, and, if pulse data has been started, any pending pulse interrupt is cleared and then the pulse interrupt is enabled. All these changes must be undone before returning to normal DOS operation and another subroutine has been written to do this. Whenever control is to be shifted to DOS, either by typing control C on the CRT terminal keyboard or because of an error condition that must be reported, this subroutine is first called to switch the peripherals to their DOS

mode. When continue is typed on the Decwriter keyboard, the first subroutine is called to switch the peripherals back to the online program mode.

The clock task then uses the current date and time words (see Table XI for the format) to calculate the (double precision) number of 1/64 hour intervals since the start of the base year, which is specified in the clock task. The number of days in the base year is also specified, so that this calculation is done correctly for two consecutive years. This permits the number of 1/64 hour intervals to be continuous over New Year (when the bubble chamber always seems to be scheduled to run), but the programmer should change the base year sometime during the second year while the chamber is not running. If the current 1/64 hour number is different than the previous one, a new 1/64 hour interval has begun and an event variable is set to read in the fast data block. The low order eight bits of the number are stored; these are the offset, relative to the start of the group for all fast data, where this new data is to be stored on the disk. Both the current and previous 1/64 hour numbers are shifted right and this process repeated for the 1/16 hour (slow data), ¼ hour (quarter hour data), and two hour (mean and standard deviations) intervals. This procedure results in the data reads beginning at even intervals based on the time of day and data storage on the disk occurring with each group being used as a circular time based buffer with the periods shown in Table IX. The first time through this code, after program start up, no event variables are set, so that all reads start at the beginning of the next preset time interval. The two hour event variable is set on program start up and the data

processing task (Section V) will examine the first partial sum block on the disk. If the date and time recorded there is within the current two hour period no action is taken, but if it is not, the means and standard deviations for the old two hour period are calculated, stored in the proper disk location, and the partial sum blocks cleared for the current two hour period.

Next, the clock task processes the timers. These are set up by any task needing an event variable set after an interval of time. The task calls a subroutine specifying the event variable address and the number of ½ second intervals to wait before setting that event variable nonzero. The subroutine first checks the event variable address against those already stored in the currently active timers. If a match is found, the specified interval is used to reset that timer. If no match is found, the first inactive timer is set to the specified interval and the event variable address stored. When the clock task is executed (every ½ second), the timer intervals are checked and counted down by one if nonzero (i.e., active). If this countdown results in the interval becoming zero, the corresponding event variable is set nonzero. If a task were waiting on this event variable, that task would then be activated by the BSX supervisor when it became the highest priority task needing service. Currently, 15 timers are available, but this could easily be increased if necessary. The timers require 106 words of memory for the code and storage; additional timers would require two more words each.

The state of six bits on the Bison interrupt and gate unit input register are then checked. These indicate the open/closed state of six bubble chamber control valves; five are valves which maintain level in cooling loop heat exchangers and the sixth is the valve

shown in Figure 1 to let liquid helium into the magnet. The number of reads and the number of times each valve was found open are recorded in memory. One of the derived quantities subroutines for the slow data block (Section V) will compute the percent of time open for each valve, store the result in the slow data block, and zero the memory locations for the next interval. Some difficulties (probably a program problem) have been experienced in always receiving the interrupts for the scanivalve and the DVM, so a timer is set after each operation that should result in such an interrupt. If a real interrupt has not been received in this time, the clock task will issue a fake one.

Finally, the clock task sends a pulse, via one bit of the Bison interrupt and gate control output register, to reset an external timing circuit. If this circuit is not reset for five seconds, it will cause a bubble chamber alarm indicating that the online program has bombed out. To encourage operators to get the mag tape logging started properly, this pulse will not be sent out until this has been done, so the bubble chamber alarm cannot be reset until the mag tape logging commands have been entered. The clock task then waits on the event variable which is set every ½ second by the clock interrupt handler. It then repeats the procedure which starts in the second paragraph of this subsection.

C. Command Task

A Qtran request[6] is issued to input a line from the CRT terminal keyboard and the command task waits until the line is entered and the return key pressed. The Bison terminal driver[7] (KBIOHT) has been written to accept output lines (to the CRT screen), even with this input request pending, until the first character of the command

is entered. Once the return key is pressed, the command is converted by the Bison routine CICONV[9]. The @ symbol required by CICONV is entered in the buffer by the command task and echoed on the CRT screen by KBIOHT when the first character of the command is typed, but does not have to be typed by the operator. This was done to cut down the number of characters that the bubble chamber operator must type for a command.

If CICONV detects an error in the command, an error message is written on the CRT screen to show what was wrong and the task waits for the next command. If the command is legal, CICONV calls the proper subroutine to carry out the desired action. Such subroutines generally make additional checks on the arguments; if some problem is found, they change one number in their argument list and the command task code will then write the proper error message on the CRT screen. Because so many of these subroutines are in the first overlay, CICONV has been slightly modified to call the first overlay into memory and reserve it before such subroutines are called and to release it after they return. Most of these subroutines require a very short time to complete the requested action. If a more lengthy job is requested, an event variable is set for the lists task and the command task returns to wait for another command (see Section VIII). Currently, 48 commands are defined, but only about 17 of them are used by the bubble chamber operators.

D. Overlay Control

Before entering code in one of the overlays, the task must put the desired overlay number into R0 and call the overlay control subroutine. The front end of this subroutine is re-entrant and, if the overlay area is busy with another task, will put the task making

the new request in the wait state until the overlay area is free. When the overlay is available, the subroutine checks the overlay segment currently in memory and reads in the requested segment unless it is already there. A flag is set to show that the overlay is busy and the subroutine returns control to the address on the top of the stack. This address was set by the task making the request and can be a location in the overlay. If the overlay control subroutine was called with the usual JSR instruction, control returns to the next instruction in the task, which may now call subroutines in the overlay segment. When the task has been finished with the overlay code, it calls a subroutine which sets the proper event variable showing that the overlay is free and returns. These overlay control subroutines require 52 words of memory.

E. General Buffer Control

Tasks requiring general buffers must first call the buffer control subroutine, specifying the number of buffers needed and address of a table for the buffer address as arguments. The front end of the buffer control subroutine is re-entrant and, if the subroutine is busy with another request or there are not enough free buffers, it will put the requesting task into the BSX wait state on the appropriate event variable. If enough buffers are currently free, the subroutine searches the buffer flags, assigning the free buffers to the requesting task by making their flags busy and transferring their start addresses to the table specified in the subroutine call, until the request is satisfied. The count of currently free buffers is then reduced by the number reserved for for the requesting task and event variables set if ≥ 1, ≥ 2 ..., ≥ 5

buffers are still free. Control is then returned to the issuing task, which then has exclusive use of the reserved buffers until they are released.

When the task has finished using the buffers, it must call the buffer control subroutine again to release the buffers, specifying the same arguments as before, except with minus the number of buffers to be released. The subroutine searches its table of buffer addresses until it matches the first entry in the address table specified in the call, marks that buffer flag free and repeats until the specified number of buffers have been marked free. The count of free buffers is increased by the number of buffers released and the event variables described above are reset. Control is then returned to the task issuing the request. The buffer control subroutine requires 120 words of memory, which includes the buffer flags and start address for 17 general buffers.

F. CRT Terminal Driver

The Bison routine KBIOHT[7] has been modified to drive the TEC 430 CRT terminal used for the online program. The CRT terminal was acquired because it is much more convenient for editing lines of information as discussed in Section IX and makes the page display described in Section VIII possible. The Decwriter was then available to make a permanent log of limit checks out of range, Section V. The improved line editing features are also available for the commands entered by the bubble chamber operators who are already familiar with an identical terminal in the bubble chamber control room, used to broadcast bubble chamber status information on the neutrino lab TV system. Finally, the CRT terminal does not generate piles of paper containing commands which are usually of little long term interest.

Aside from changing the address of the hardware registers from

those for the Decwriter to those for the CRT terminal, it was also necessary to make changes to the driver to allow the improved line editing features. An end of the line buffer pointer and a counter of the total characters on the line were added. To output a line of text for editing, the line is formatted as usual with FMTPUT[10] but then the line feed character is removed and the count of characters put into the total character counter. The line is then written on the CRT screen in the usual manner. An input request is given to the CRT keyboard using the same buffer as was used for the output. KBIOHT was modified so that this buffer would not be cleared by the input request if the total character counter was non zero. KBIOHT then adds the total character counter to the initial buffer pointer for the end of buffer pointer. The character count and buffer pointer originally in KBIOHT are used to indicate the present position of the cursor on the CRT screen. A test was added to the keyboard interrupt handler in KBIOHT to first check the input character and branch to a special code if it was a line editing character.

The character to move the cursor forward or backward one space will now increment or decrement the character count and buffer pointer, be echoed on the CRT screen to move the cursor, but will not be stored in the buffer. Attempts to move the cursor to before the first character or after the last one are ignored. Normal characters typed with the cursor in the middle of the line change that character in the buffer, echo the character to make the change on the CRT screen to the position over the cursor, increment the buffer pointer and byte count, and advance the cursor one space. Characters added to the end of the line do the above as well as

increasing the total character counter and end of line pointer. Delete or insert character codes result in an echo of that code to make the change on the CRT screen, the movement backward or forward of the characters in the buffer between the buffer pointer and the end of the line pointer with the corresponding adjustment to the total character count and end of the line pointer, and insertion of a blank or removal of a character at the buffer pointer position. The erase to end of line key is also enabled; this is done by echoing the control character to make the change on the CRT screen, setting the total character count equal to the character count, and setting the end of buffer pointer equal to the buffer pointer. Rubout results in the backspace and then delete character operations described above. Other editing characters (used for page editing) are ignored.

When the return key is pressed, the space forward operations are repeated until the cursor is positioned just after the last character. The usual KBI0HT carriage return procedures are carried out to insert a carriage return and line feed in the buffer and notify the requesting program that the line input operation is complete. Note that this results in the entire line being transmitted, no matter where the cursor is positioned when the return key is pressed. If the line feed key is pressed, the erase to end of line operations are done first and then the control character is treated as if it were a carriage return. Control U results in moving the cursor to the start of the line, erasing the line on the CRT screen, clearing the input buffer, clearing both character counters, and setting both buffer pointers to the start of the input buffer. This follows the usual DOS convention.

If KBI0HT is entered with the total character count equal to

zero, the input buffer is cleared and both the buffer pointer and end of buffer pointers set to the start of the buffer. When the first character is entered from the keyboard, the CRT screen from the cursor position to the end of the page is erased to clear out any characters that may be displayed there (which would tend to confuse the operator). Once characters have been entered on the keyboard, the line editing features described above are available if needed.

G. Changes Needed to BSX

BSX will return control to the DOS monitor if one of a number of error conditions are found or a control C is typed on the keyboard. It is important that DOS will be able to run correctly in this case, either so that the online program can be continued after certain error conditions (i.e. a device not ready) are corrected, or so the online program can be restarted easily. Some features of DOS are useful in examining the state of the online program after certain errors, which is frequently of use in debugging new versions of the program. DOS will not run correctly unless the clock is returned to its DOS mode of operation and the decwriter keyboard enabled. (BSX already resets the interrupt vectors it modified when online program execution began). BXCODE was modified to call the subroutine to return peripherals to their DOS mode of operation before control is returned to DOS, and to call the subroutine to set the peripherals up for the online program again if the operator commands DOS to continue the online program. These two subroutines are described in Section XB, clock task.

The DOS driver for the memory scope, in DOS version 9, issues a "VT not ready" message whenever the form feed character is sent to the scope to start a new page. This is done to permit the

operator to read the previous page and then type continue when he wants the new page. This online program saves pages on the memory scope for 10 seconds by other means (see Section VIII) and the above procedure, which halts online program execution and switches to DOS, is not acceptable for an online program. BXCODE was modified to check error messages before switching to DOS and if the message was "VT not ready" the error is ignored. A second such error with ½ second is allowed to go through, because it would indicate a real problem with the memory scope.

H. Memory Use

Table XIII gives a list of the bubble chamber online program source files, including their functions and memory requirements. Programs written specifically for the bubble chamber, Bison programs written by the computing group at Fermilab, and DEC supplied programs are listed separately. The total memory requirement is larger than the memory on the computer; this is possible because the three overlay segments share the same area in memory.

Table XIV shows the breakdown of memory use for different jobs when the online program is running without ODT. Four general buffers are not absolutely required, so 1583 words are available for program expansion. Of these, I have reserved 128 words for additional monitor buffers; perhaps this number could be reduced if required. Of course, additional overlay segments can be added which require little or no additional memory.

XI. UNUSUAL PROBLEMS

A. Software

In 1974, when the first version of the online program was written, minor changes had to be made to many of the BSX source files because a newer version of the MACRO[3] program was used in the DOS system for the bubble chamber computer. The problem was that the new MACRO program did not consider .CSECT names as .GLOBL definitions and I believe this error has been corrected in the current Bison library tapes.

An amusing error was discovered in the Bison CKINTR[8] routine which caused the online program to bombout at midnight on the computer's clock. The bubble chamber computer was nicknamed Cinderella[14] because of this error. It occurred because CKINTR did not actually change the date word until about seven minutes after midnight and a fortran call for the current data and time during this period would cause the program to bombout. Another problem was that CKINTR lost 22/60 of a second every 9.11 minutes, so the computer clock appeared to lose about one minute per day. Both of these errors were corrected and a copy of the corrected CKINTR given to the Computing Department.

The Bison subroutine ISQRT[13], which calculates square roots of integer double precision numbers was found to go into a loop if given a number greater than or equal to 268,435,456 (i.e. 2^{28}). This was solved by testing the number before calling ISQRT and if it was greater than or equal to the above value, putting the error code for data too big in the answer and not calling ISQRT. Since the online program only uses ISQRT to calculate standard deviations, numbers this large are of little interest.

When overlays were first attempted, the DOS monitor would bomb-out when trying to load the online program load module. After almost a week's worth of effort, it was discovered that DEC had published a patch to DOS version 9 (then in use at the bubble chamber computer) to correct this problem, which only occurred when trying to load programs longer than 16K from contiguous files (which are necessary for overlayed programs). After the patch was made, DOS version 9 worked properly and the bubble chamber computer has since been shifted to version 10 which does not have this problem.

As discussed in Section VIII, the Bison PLOTB[5] program to make graphs on the line printer did not work correctly. The problem was found to be either in the Computing Department plotter driver or the plotter direct memory transfer interface on PDP-11/20 computers. I solved the problem by using direct commands to the hardware registers to output the plot data. I believe that the Computing Department is working on a longer term solution to this problem.

B. Hardware

* Most hardware problems on the bubble chamber computer have been rather obvious and, after the problem is reported, the DEC repairmen can find the trouble and fix it quickly with little or no input from the programmer. However, about once a year, a problem develops which requires perhaps a week of effort by the programmer to prove that it is caused by hardware and then another week or two of the programmer's effort working with the DEC repairman to get the problem localized and repaired. I would expect that any small computer will require two or three weeks per year of effort

by an experienced programmer to diagnose and localize such problems. The Computing Department personnel can frequently offer valuable advice, but they seem to have little time available to work on such problems. Most of the time must come from the programmer assigned to the machine.

The first problem occurred shortly after the computer was installed. The computer would randomly bombout, even when running DOS systems programs, which should be well debugged. Intermittent problems are very hard to locate, and the DEC repairmen tended to blame noise on the AC power line. This cause was eliminated by borrowing several AC power filters and using them in the computer's power line. When the bombouts continued, even with the computer on filtered power, the repairman seriously went to work and finally fixed the problem. I must confess that I don't remember which component was a fault, but after the repairs were made, the filters were removed and the problem did not return.

The next problem occurred almost a year later. When a block of data was written from memory to the disk, sometimes a few words in the block in memory were zeroed. When the usual DEC diagnostic programs failed to detect the problem, I wrote a short test program specifically to test for this error. After this, the DEC repairman found an obscure DEC diagnostic which would also detect the error. The cause remained difficult to locate; at one point the error would not occur if the expansion box was pulled out, but failures occurred when the expansion was returned to its normal position. Some unibus cables were replaced and better insulation put between the wire wrap pins and the expansion box top cover and the problem went away. We are not sure what the actual cause of the

problem was.

The latest problem occurred last spring with the magnetic tape unit. Fatal error messages and program bombouts would occur when the tape was commanded to backspace. Occasionally, it would backspace to the beginning. Two hardware problems were finally found. A faulty capstan motor was not always running at constant speed and the tape unit electronics did not have up-to-date field changes. This latter problem was found by swapping electronics with another mag tape unit. Considerable programmer effort was required to test various combinations and fixes, since the normal DEC diagnostics would not detect the problem. After the motor was replaced and the electronics fixed, the errors no longer occurred.

XII. FUTURE POSSIBILITIES

* Although the bubble chamber online computer system is now essentially complete and has become more and more valuable for bubble chamber operations during the last two years, one can always think of new features that could be added and would further improve bubble chamber operations. The addition of new lists, the read in of new data points, and the calculation of new averages, etc., can be done without changing the program at all, but by modifying the control blocks on the disk, as described in previous sections. As can be seen from figures 6 and 7, there is room for 95 more pieces of data in the slow data block and 143 more in the fast data block. There are many extensions possible to the few variable summary feature described in Section VIII and some of these are outlined below. Using the computer for control functions and several other possible new features are also discussed. Which of these are actually done depends on the needs of the bubble chamber operating crew and the support given to this project by the Laboratory.

The few variable summary (Figure 11), which gives the values of a few variables at a large number of successive times could be expanded to give maximum and minimum values, totals, and means and standard deviations of each variable during the time period covered by the summary. Much of the code required for this already exists in the program. A straight line least squares fit for each variable would give its average rate of change during the time period of the summary. A straight line fit of one variable vs. another would save much of the programmer's

time when, for example, calibrating the dynamic chamber pressure transducers against the precision static pressure transducer. Histograms of the data in the summary and the selection of which data is used for a display, depending on the value of another variable, would give, for example, separate histograms of the number of hadron beam tracks for the different pulse numbers when multipulsing. Such information would permit rapid feedback to the beam line and the accelerator and improve picture taking efficiency. It might be useful to calculate a new variable from two or more of the variables in the summary and be able to make displays of this new variable. The ability to delete unwanted entries from the summary before the displays or calculations are made would be of use. A more difficult project would be to set up a special, more frequent read in of a few variables and list or display them using the existing features for the few variable summary data. This might be useful, for example, to study the pressures in various vessels and the control valve positions with good time resolution during bubble chamber pressure test. It would be convenient to make it possible to automatically update any specific graph every time new data was read in. Since most of the code for these projects could be in any overlay segment, there is no basic difficulty in doing them, if enough programming effort is allocated to the project.

Some additional programming effort on the graph displays might be worthwhile. The Bison plotting routines could be modified to use the format for the tic mark labels (PLOTA[12]) or the Y coordinate values (PLOTB[5]) which inserts the decimal point in the proper place (see Section VIII). Both routines already use

FMTPUT[10] to format these numbers, but some study of these rather long programs would be necessary before this could be done properly. More such effort could result in date/time labels on the time axis of these graphs instead of the "intervals ago" labeling done presently (see Figures 12, 13, 14, 16 and 17). An effort is in progress to select intelligently rounded intervals for the graphs instead of simply using the maximum and minimum data values as is done presently. Both PLOTA and PLOTB are advertised to have this feature as an option, but neither one does it well enough to use.

Page displays on the CRT screen for slow and fast data, similar to the existing feature for pulse data, could be added without too much programming effort. Because of the slower read in rate for these data, the initial request for such a page should read from the disk and display the last five readings for the selected data. Otherwise the operator would have to wait too long to get useful rate information.

At present, the operator must specify a particular piece of data by its data block and data point number, see Sections VIII and IX. Usually he must look this up in the read lists, Figures 6 and 7. A dictionary look up routine could be written which would accept the two or three letter plus number bubble chamber naming convention, search the slow and fast data ID blocks on the disk, and find the required block and data point number. Such a routine would require a few weeks of programming effort, but would save some operator time.

The present time of the pressure minimum inside the bubble

chamber is currently just the time of the minimum pressure reading from the selected transducer. Since this minimum is rather broad and there is some noise on the signal, even after a filtering circuit, a better estimate of the time of pressure minimum could be obtained from a curve fit to pressure readings taken near beam time. These readings are stored in the pulse input buffers, see Section VI, but several weeks of effort would be needed to write and debug the fit subroutine using multiple precision integer arithmetic. If this is done, it would be almost necessary to be able to display the pressure data points and the fitted curve on the memory scope to check that the fit was being done correctly.

Some bubble chamber data varies rather rapidly and sampling it about once a minute for the fast data block does not really give a good idea of its short term behavior. Such data could be read every ½ second by the clock task, just as the position of the cooling loop valves is already done, see Section XB, and the short term average stored in either the slow or the fast data block in the same way. Digital input or the fast A/D converter are the only devices on the system fast enough to input such data. The gauge which measures the return flow from the bubble chamber cooling loops is the most obvious example of an indicator which should receive such treatment. Few memory locations and only a small programming effort would be required to do this job.

* The intensity of the proton beam on the target for Neutrino experiments vs. the event rate in bubble chamber test strips has been frequently very important in monitoring the performance of the Neutrino beam line. The EMI computer records this intensity, together with the bubble chamber picture roll-frame number, but

presently is not able to give an online listing of these data. In practice, the experimenters record the intensity by hand from the MAC beam line computer display. If this intensity were recorded by the bubble chamber computer in the pulse data blocks, the few variable summary feature of the online program could provide such a listing and save the experimenter their hand recording job. Transmitting the beam on target intensity signal from the MAC system could require another CAMAC crate and modules which are rather expensive and new procedures to be learned by the bubble chamber operators to set up the MAC system data transfer correctly. Also, the signal arrives rather late after the pulse and handling the data correctly would require considerable re-programming for the bubble chamber online program. The transmission of the signal on a TV sound channel[15], already developed for other uses at the bubble chamber, would be a cheaper and easier method to get this signal into the bubble chamber control roon, where the computer could read it in with the fast A/D converter. The hadron beam counts into the chamber for each pulse are already read into the computer, so such lists could currently be made for hadron experiments.

As discussed in Sections V and X, the current date and time are used to label all data read into the computer. The slow and fast data blocks as well as all the means and standard deviation blocks are stored on the disk in locations depending on the value of this date and time. Therefore, it is vital that the correct date and time be entered into the computer each time that it rebooted. Errors in the past have caused complications and the loss of some data on the disk. Efforts are currently underway to read the control

room digital clock during the online program start up phase, and to reset the computer date-time words from this clock. In addition to reducing the possibility of incorrect dates or times, this will also speed up the process of restarting the computer.

* It is expected that additional subroutines will be written to calculate new derived quantities for the slow and fast data blocks (Section V). Since data locations and constants are stored in the control block(s) on the disk, very little memory space is required for such subroutines. One such subroutine could convert the level of a cryogenic liquid in a storage dewar, usually read as inches of water, into volume. Another could calculate compression ratios for each stage of the compressors in the bubble chamber support systems. When all the necessary data has been interfaced into the computer, these compression ratios would give an early indication of ring or valve problems in the compressor. Another such subroutine could be used to convert the nonlinear readings from vacuum gauges into linear pressures. A subroutine to calculate the volume of gas, in a tank, under standard conditions, from the temperature and pressure would be very useful. When the required data has been interfaced, an accurate inventory of helium in the system could be kept by the computer. This would give an early indication of losses and could result in reduced helium purchases. I expect that other such applications will be found for the online computer.

Currently, when the magnetic tape unit is cleaned or a tape fills up and must be rewound and changed, about ten minutes of data is not recorded for long term storage. However, this data is stored in various locations on the disk. With some programming

effort, the computer could keep track of which disk blocks have not been recorded on tape. When tape logging was resumed, these blocks could be read (one at a time) from the disk into the general buffer and then written on the tape. Once this feature becomes operational, further programming effort could give the bubble chamber operator online access to all the data stored on the magnetic tapes. With the chamber pulsing twice per accelerator cycle, almost 1½ hours worth of pulse data is stored on the disk. During this time, the tape could be changed, if necessary, and repositioned to the start of the interesting data. The operator would then specify the type of data desired, and the computer would read the tape, select the specified data, and write it in the currently unused 256 block group on the disk. The tape would then be repositioned and data logging resumed, with the unlogged blocks from the disk going on the tape first. With some modification, the present display routines would then be used to study this special data. Up to a few months of programming effort would be required to do all this, and some careful consideration of actual experience with the online program should be made before deciding whether or not to go ahead with this feature.

It would require far less programming effort and less operator time to add a second RK05 disk drive to the system and use it to store additional bubble chamber data, particularly the all fast and all pulse data types. Another possible hardware addition would be a second multiplexed digital volt meter. The DVM is more accurate than the fast A/D and because of its several ranges, no amplifier is required for each signal. Because of the rather

slow settling time of a DVM, it would be better to add a second system, rather than just increasing the number of multiplexed inputs on the present system. The second DVM could probably eliminate the lower ranges of the present DVM (see Table V), and be made somewhat faster. In the longer range future, it may be necessary to add additional digital input; it should match the 16 bit PDP-11 words to reduce program complexity (memory required) and not be CAMAC, since those modules are rather expensive.

No offline analysis program for the bubble chamber data tapes has been written yet. This should not be too big a job, because it can be done in Fortran on one of the large Fermilab computers. The Computing Department already has subroutines to read tapes from PDP-11 computers. There are also extensive display programs, such as SUMX or KIOWA, in use on these computers. Putting these elements together should not require more than several weeks of programming effort.

For the reasons given in Section IV, the computer system does not have any control functions as yet. During bubble chamber runs, when the control room is always manned, some control functions may be an aid to operator. Between bubble chamber runs, only a two man crew is present at the chamber and it would be of real value to reduce the time they must spend in the control room; thus freeing them for maintenance and development jobs. During these between run periods, large quantities of expensive neon-hydrogen or deuterium liquids are stored in dewars. Liquid hydrogen is transferred to condensers in these dewars and allowed to boil off to compensate for the heat leaks into the dewars.

Currently, the hydrogen comes from a storage dewar, and the air controllers that operate the transfer valves sometimes do the job rather inefficiently, resulting in a greater use of liquid hydrogen than is absolutely necessary. Using the online computer to control these transfers could potentially save liquid hydrogen and hence money.

* The large hydrogen refrigerator, used when the bubble chamber is running, is much too big to operate for only the small amount of liquid hydrogen needed to keep the storage dewars cold. The laboratory may acquire a much smaller refrigerator for this job. If this is done, it should be as automatic as possible, and using the online computer system as a part of the control system for it should be considered, starting with the earliest planning stages.

During running conditions, one worthwhile control function for the computer would be to regulate the chamber temperature. The heat load on the chamber, which must be removed by the cooling loops, consists of a constant load and the heat caused by pulsing the chamber. Because of the large mass of liquid in the chamber, changes in temperature caused by changes in the pulse rate or depth do not show up on the vapor pressure thermometers until several tens of minutes after the change occurs. In the past, better chamber temperature stability has been achieved by the operator manually changing the cooling loop control set points on the air controllers when the pulse heat load is varied. Better temperature control results in better physics pictures, because bubble size and density are more uniform and because schlieren effects are minimized if there are no periods of excessive chamber cooling. The computer already measures the pulse heat load, so it

would be reasonably easy for it to use this information from the previous measurement intervals plus the difference between the current average chamber temperature and the desired temperature to automatically enter new set points into the air controllers for the next time interval. The time interval for the slow data reads (1/16 hour) would be just about right for this operation. The only new hardware required would be electrical to pneumatic converters for the set points and the programming changes would not be large, although some experimenting would be required to get the proper control equations.

Another application of computer control would be to adjust the expansion system drive gas pressure to get the desired pressure drop during the pulse. Unlike the above case, the chamber pressure drop appears to remain constant if the expansion system control pressures remain constant. Therefore, I believe that what is really needed here are better pressure regulators on the expansion system, rather than computer control. The efficiency of the hydrogen refrigerator is improved slightly if the split of the warm compressor discharge gas to two heat exchangers is adjusted so that the low pressure hydrogen return and nitrogen vent gas temperatures are equal. The computer reads in both temperatures with the copper constantan thermocouple system, and could use this information to control the valve which splits the flow. However, experience shows that, once this valve is set properly, it need not be adjusted unless drastic changes are made in the refrigerator operating conditions. The option of calculating the temperature difference between the two thermocouples, setting up a limit check and alarm on this difference,

and letting the operator adjust the valve himself would be nearly as effective and would require no program changes at all.

* This section on future possibilities is certainly not a complete list of all that could be done with the computer system and is only intended to be a rough guide for the immediate future. I am sure that continued experience with the computer under actual running conditions will add new items to this list and that practical considerations will mean that some of the items will never be done.

XIII. CONCLUSION

* The online computer system described in this note has successfully met the need at the 15' bubble chamber for accurate data logging and display of this information. Knowledge of the time rate of change of various parameters is vital at a bubble chamber, and the system has been designed to provide this knowledge. When the computer system has been particularly successful in meeting some bubble chamber need, a short description appears at the appropriate position in the text, and these have been flagged for general interest. Since bubble chamber operators are, in general, unfamiliar with computers, a special effort has been made to minimize and simplify the actions required by the operator for proper computer operation. A major problem with any small computer is to fit everything required into the restricted amount of memory available. This has been solved by making extensive use of the disk and by coding the entire program in assembly language. Much of the time invested in the program was needed to accomplish this, and a large part of this note has been used to describe these efforts in detail.

* The computer system periodically reads in data about the bubble chamber, converts this data to physical units, and calculates additional data which depends on one or more of these data points. The present program will handle 512 such pieces of information. During every chamber pulse, data is read and processed by the computer. This results in an additional 125 pieces of information. All this data is saved on the disk for various lengths of time and written on magnetic tape for long term storage and offline analysis. Almost one million words of information on the disk are available

to the bubble chamber operator for displays on command. These displays may be lists or graphs; all of which are set up to include the time rate of change of the selected parameters.

* I believe that some of the techniques, and perhaps even the actual code, developed for the bubble chamber online program, would be of use in other small computer projects at the Laboratory. A list of the most likely candidates follows, with the section reference in parenthesis.

1. CRT terminal driver (X F)
2. Timers (X B)
3. General buffers (X E)
4. Control and constants tables on the disk (V)
5. Editing of blocks on the disk (IX)
6. List specifications on the disk (VIII)
7. Start up code in an area which is then used as a general buffer (X A)
8. Magnetic tape positioning and labeling (VII)
9. Insertion of decimal points into integer numbers for output (VIII)
10. Error codes stored in place of data (VIII and Table XII)
11. Handling data as systematically as possible (V)

* Much of the bubble chamber online program is, in fact, a general data logging and display program. With rather minor modifications it could be used for similar applications at the experiment areas for beam line data, at the new helium liquifier plant, or perhaps at the accelerator. Tables I and III list what hardware would be required, but data input devices would be rather different for most of these possible applications. In fact, most of these would

probably use a data transfer from an existing computer system (i.e., the beam line MAC systems).

Some information about my programming experience with this system may be of interest to anyone planning a small computer project. Because of the memory limitations discussed in Section IV, the final version of the program is written entirely in assembly language. A novice assembly language programmer, even one with considerable Fortran experience, will require three to six months of full time practice to become reasonably proficient (i.e. greater than 50% of his potential rate) in writing and debugging such code. If he spends less than half time on this, the initial period will tend toward the upper limit of six man months. One estimate is that a good, experienced assembly language programmer will generate only three to five lines of code per hour. This estimate includes the time to plan the program, write the instructions, enter them into the computer, and debug the program. Code which is not used in the final program is not included in the yield.

* I estimate that I have spent 15 man months on the program, spread out over a two year period. Deducting three months of learning experience leaves 2,000 hours of productive programming. The programs listed in Table XIIIA require 16,000 words of memory, so I managed to fill eight words per hour. PDP-11 instructions require one, two, or three words of memory with the average between 1.5 and two. This gives a net yield of four to five instructions per hour. The programs in Table XIIIA contain 11,000 lines. Perhaps 10% of these are comments and other non-code generating lines, so this estimate also gives five lines of

code per hour.

* Depending on ones other duties and how much already existing code can be used, programming a small computer can easily take two or three calendar years. Because of the learning curve and the need to write certain general routines at the beginning, the usefulness of the programming effort during the first year can appear quite small. Some Fortran programs can be used during this period to make the computer do a few jobs, and this was done on the bubble chamber system. Because of memory limitations, the functions of these Fortran subroutines were later coded in assembly language. I would be happy to discuss interesting aspects of the bubble chamber system with anyone planning a small computer project at the Laboratory, and I urge any Laboratory Physicist who does not have a permanent appointment to talk to me before committing himself to such a project.

Hsi Feng has been responsible for much of the hardware, from planning through the commissioning of the special hardware devices (Table III), which has been vital to the success of this project. Charles Mangene has done much of the work necessary to interface the special devices to the computer. Many members of the bubble chamber operating crew have helped to interface bubble chamber data to the computer and contributed valuable suggestions. Jim Early has assisted on several of the recent program improvements and has taken over the computer system.

REFERENCES

1. "PDP 11/20 Processor Handbook", DEC, 1971

2. "PDP 11 Peripherals Handbook", DEC, 1973. BISON Documentation. Fermilab Computing Department.

3. "DOS/BATCH Handbook", DEC-11-ODBHA-A-D, December, 1974

4. "BISON Display Hardcopy", Bison PN-19.1 Fermilab Computing Department

5. "PLOTB-Versatec Plotting Package", Bison PN-48

6. "BSXD User Manual", Bison PN-4.1 (including appendices)

7. "Teletype Driver", Bison PN-23

8. "Minimal Routines for the KWIP Programmable Clock", Bison PN-43

9. "Command Interpreter", Bison PN-11.2

10. "ASCII Output String Formatter Subroutine", Bison PN-33.1

11. "Kinetic Systems KS0011 CAMAC Branch Driver Handler", Bison PN-12.1

12. "Bison Plotting Package", Bison PN-2.5

13. "Integer Square Root Function", Bison PN-25.1

14. I believe that the second, more general meaning of Cinderella also applies to the bubble chamber online computer system.

15. "An Analog Signal Transmission and Distribution Method", H. Feng and C. Mangene, TM-675 Fermilab, July, 1976

* TABLE I - HARDWARE SUPPLIED BY FERMILAB COMPUTING DEPARTMENT

MANUFACTURER	MODEL	DESCRIPTION
DEC	PDP 11/20	Central Processor
DEC		Extended Arithmetic Element
DEC	KW11-P	Programmable Real-Time Clock
DEC	BM792-YB	Auto Loader
DEC		8K Memory
Datacraft		20K Memory
DEC	LA30	Decwriter terminal 30 characters/ second
DEC	TC11	Dec tape controller
DEC	TU56	Dual Dec tape drives
DEC	TM11-A	Magnetic tape controller
DEC	TU10-EA	Magnetic tape transport, 9 track, 45 ips, 800 bpi
DEC	RK11-CA	Disk controller
DEC	RK05-AA	1.2 M word cartridge disk drive
Versatec	200A	Printer/Plotter 600 lines/minute
DEC	AA11A	Tektronix scope control
DEC	AA11D	D/A subsystem for above
DEC	BA614	D/A converter (2) for above
Tektronix	613	Storage scope
Kinetic Systems	KS0011	CAMAC Crate Interface
Fermilab	-	Bison Interrupt and Gate Control
DEC	DR11-C	General purpose interface for above

TABLE II - STORAGE DEVICES

Blocks*	DEVICE
112	Memory
4,800	Disk (RK11, RK05)
1,156	Two Dectapes (TC11, TU56)
∿21,000	Magnetic Tape (TM11, TU10)

* 1 block = 256 (16 bit) words

* TABLE III SPECIALIZED HARDWARE ADDED FOR THE 15" BUBBLE CHAMBER COMPUTER SYSTEM

MANUFACTURER	MODEL	DESCRIPTION
Datel Systems	System 256	Fast A/D with 128 differential inputs (inc. 8 samples & hold) and 8 D/A outputs
Datel Systems	2561-PDP-11-1	Register interface for above
Datel Systems	2561-PDP-11-2	Direct Memory transfer interface for above
TEC	430	CRT Terminal
DEC	DL11E	Interface for above
Data Precision	3500	Digital Voltmeter, 5 1/2 digits
Fermilab	--	Multiplexed input for above
DEC	DR11-C	Interface for above
Scanivalve	SSS 64 CBM	Pneumatic Multiplexer for 3-15 psi signals
Standard Engineering	1410	CAMAC Crate
Joerger	OR	CAMAC 48 Bit digital output module (4)
Joerger	IR	CAMAC 48 Bit digital input module (8)
Lecroy	2550B	CAMAC quad 100 MHz scalar module (2)
Jordway	70A A1	CAMAC crate controller
Fermilab	--	Computer controlled addressing for the gold chromel thermocouple system
Fermilab	--	Computer controlled addressing for the copper constantan thermocouple system

* TABLE IV GENERAL DATA DEVICES

DEVICE NUMBER	DESCRIPTION	POSSIBLE SUBADDRESSES
1	Gold Chromel Thermocouple System	100
2	Copper Constantan Thermocouple System	100
3	Digital volt meter	64
4	Reserved for future DVM use	---
5	Fast Analog to Digital Converter	128
6	Air logic system Scanivalve	64
7	Not Used	---
8	CAMAC digital input	see text

TABLE V DIGITAL VOLTMETER RANGES

DVM SUBADDRESS (octal)	LEAST COUNT (μv)	FULL SCALE (volts)
0XX	1000	± 11.999
2XX	100	3.2767
3XX	10	0.32767
1XX	1	0.032767

TABLE VI SLOW AND FAST DATA STATUS BYTE CODES

DATA STATUS BYTE	RESULT
0	Data point is not read in (no data error code is put in its place)
1	Data point is read in but no conversion to physical units is done. The raw data value is transferred to the final buffer, but it is not included in the futher analysis (derived quantities, partial sums or limit checks)
2	All normal processing steps are done on the data point, except no limit checks are made.
3	All processing steps are done on the data point.

TABLE VII DISK BLOCKS USED TO DEFINE ONE BLOCK OF GENERAL DATA

# OF BLOCKS	USE	STRUCTURE(words used for each data point)
1	Read in control list	High byte= device number (Table IV) low byte = subaddress
4	Convert raw data to physical units	Control word,b,s,a (equation 1)
1 +	Calculate derived quantities	String for each operation; subroutine number, output location, input locations or constants, Ø.
6	Store partial sums	N (1 word), Σ x (2 words), Σ x^2 (3 words).
9	Limit Checks	Control word, 4 sets of lower and upper limits.
6	ID and format	Format, label number, label name (4 ASC II char.), units (4 ASC II char.)
Total 27 +		

TABLE VIII EVENTS DURING CHAMBER PULSE FOR WHICH TIMING IS RECORDED

BIT	EVENT
0	Beam for A (Hadron) Experiment
1	Beam for B (Neutrino) Experiment
2	Light Flash Trigger
3	Data Box Trigger
4	Camera Trigger
5	Latch Valve open/closed
6	Recompression Valve open/closed
7	Expansion Valve closed (Open is start of timing measurement).

* TABLE IX ORGANIZATION OF THE DISK FILE BCDATA

RELATIVE BLOCK NUMBERS	USE	TYPE	DATA STORED	DATA RETAINED
0-255	Control Tables, etc. (See Table X)	--	--	--
256-511	UNUSED	--	--	--
512-767	Means, slow data	M	2 hrs.	512 hrs.
768-1023	Std. Dev., slow data	M	2 hrs.	512 hrs.
1024-1279	Means, fast data	M	2 hrs.	512 hrs.
1280-1535	Std. Dev., fast data	M	2 hrs.	512 hrs.
1536-1791	Means and Std. Dev. Pulse # 0	M	2 hrs.	512 hrs.
1792-2047	Means and Std. Dev. Pulse # 1	M	2 hrs.	512 hrs.
2048-2303	Means and Std. Dev. Pulse # 2	M	2 hrs.	512 hrs.
2304-2559	Means and Std. Dev. Pulse # 3	M	2 hrs.	512 hrs.
2560-2815	Slow Data	Q	1/4 hrs.	64 hrs.
2816-3071	Fast Data	Q	1/4 hrs.	64 hrs.
3072-3327	Slow Data	A	1/16 hrs.	16 hrs.
3328-3583	Fast Data	A	1/64 hrs.	4 hrs.
3584-4095	Pulse Data	A	PULSE	1024 Pulses

TABLE X ONLINE PROGRAM CONTROL TABLES, ETC STORED ON THE DISK

BLOCK NUMBER(s)	NUMBER OF BLOCKS	DESCRIPTION - FUNCTION
ø	1	Latest general constants (read in during program start up)
1-9	9	Other versions of general constants
10	1	Latest general data status bytes (read in during program start up)
11-20	10	Unused, intended for other versions of status bytes
21	1	Slow data read in control list
22-25	4	Conversion of raw slow data to physical units
26	1	First block of control strings to calculate slow data block derived quantities
27-32	6	Partial sums to calculate slow data means and standard deviations
33-41	9	Slow Data limit checks
42-47	6	Slow Data ID and Formats
48-50	3	Unused
51	1	Fast data read in control list
52-55	4	Conversion of raw fast data to physical units
56	1	First block of control strings to calculate fast data block derived quantities

Table X Con't

BLOCK NUMBER(s)	NUMBER OF BLOCKS	DESCRIPTION-- FUNCTION
57-62	6	Partial sums to calculate fast data means and standard deviations
63-71	9	Fast Data limit checks
72-77	6	Fast Data ID and formats
78-80	3	Unused
81-92	12	Partial sums to calculate pulse data means and standard deviations, three blocks each for pulse numbers 0,1,2, and 3.
93-99	7	Unused
100	1	Contents of CRT screen pulse display pages
101-105	5	Pulse data titles and formats
106-129	24	Unused
130	1	Few variable summary parameters
131-142	12	Few variable summary data
143-185	43	Unused
186-199	14	List specification continuation blocks
200-249	50	First list specification blocks
250-255	6	Unused

TABLE XI ORGANIZATION OF THE 256 WORD BLOCKS

A. All blocks contain:

1. Word 0

   a). Bits 15-12, Code identifying type of information

Code (Octal)	Meaning
00	Control tables, constants tables, etc.
06	Means and Std. Dev. of slow data
07	Slow data
10	Few variable summary data
12	Means and Std. Dev. of fast data
13	Fast data
16	Means and Std. Dev. of pulse data
17	Pulse data

   b). Bits 11-0. Block number, relative to the start of BCDATA, where the information is stored on the disk.

2. Word 1. Date = (Year - 1970)* 1000 + day of year

3. Word 2. High order time word

4. Word 3. Low order time word, where seconds since midnight $= \frac{1}{60} \left[HTW * 32{,}768 + LTW \right]$

5. Unless otherwise specified the remaining words contain the data, control tables,etc.

Table XI Con't

B. Pulse Data

1. Words 4-128. Data from one pulse

2. Words 129,130. Time of next pulse

3. Words 131-255. Data from next pulse

C. Means and Std. Dev. of Pulse Data

1. Words 4-128. Means

2. Words 131-255. Std. Dev.

D. Control String to Calculate Derived Quantities and List Specifications

1. Word 4. Link to next block

a). Nonzero. Block number, relative to start of BCDATA, to use next

b). Zero. This is the last block to use for this operation

2. Words 248-255. For the first list specification block only, these words contain data block codes specifying what data is to be used.

TABLE XII ERROR CODES

ERROR CODE	MEANING
*0	No data. Data was not read or inputs for the calculation were not available
*1	Data was too big
*2	Data was too small
*3	Bad data from device
*4	CAMAC error
*5	Computer did not have control of device subaddress.
*6	Read in overrun. Not enough time to read data
*7	Control table error

TABLE XIII ONLINE PROGRAM SOURCE FILES

A. Programs Written by the Author Specifically for the Bubble Chamber System

	FILE NAME	MEMORY WORDS	FUNCTION
1.	BCTAB.MAC	957	BSX task table, clock and command tasks
2.	AUTC.MAC	285	Driver to read the gold chromel thermocouple system
3.	CKS.MAC	42	Prevents more that one task from using KS0011, repeats operation if KS0011 interrupted
4.	OUTCON.MAC	679	Control of output to alphanumeric line devices
5.	PDP.MAC	697	Task for real time display of pulse data, control of general buffers
6.	MTR.MAC	233	Portion of Mag tape control task in main segment
7.	SDR.MAC	723	Task to read in slow data block
8.	LISTS.MAC	591	Portion of output list task in main segment
9.	DATAP.MAC	1,238	Task for analysis of slow and fast data; calculates means and std. dev. for all data blocks every two hours
10.	DEV.MAC	703	Drivers for DVM, fast A/D CAMAC digital input and Scanivalve

TABLE XIII Con't

	FILE NAME	MEMORY WORDS	FUNCTION
11.	PAP.MAC	1,882	Pulse interrupt handler, reads pulse data
12.	PWP.MAC	579	Task to output pulse data to disk. Also contains the special start up code whose memory is then used as a general buffer
13.	DFI0HT.MAC	136	Driver to output (only) on the Decwriter, derived from KBI0HT.
14.	D1.MAC	148	Disk input-output control subroutine, overlay control
15.	CUTC.MAC	264	Driver to read the copper constantan thermocouple system
16.	FDR.MAC	544	Task to read in fast data block
17.	DKEDIT.MAC	1,799	Editing of control and constants tables on the disk, set up of pulse display pages on the CRT (in overlay 1)
18.	CMDS.MAC	1,185	Misc. commands, remainder of mag tape control task (in overlay 1)
19.	LOV.MAC	1,137	Remainder of list task (in overlay 1)
20.	FVS.MAC	1,700	Makes the few variable summaries, set up calls to PLOTA for graphs on memory scope, and calls HCOPY (in overlay 2)

TABLE XIII Con't

	FILE NAME	MEMORY WORDS	FUNCTION
21.	GPRINT.MAC	685	Sets up calls to PLOTB for graphs on the line printer (in overlay 3)
	B. Bison Programs Which Have Been Moderately Modified for the Bubble Chamber Online Program		
22.	KBI0HT.MAC	694	Driver for CRT terminal[7]
23.	FMTPUT.MAC	471	Formatter for line alapha-numeric output[10]
	C. Bison Programs Used With Little or no Modification		
24.	BXSCAN,etc	1,370	BSX supervisor, including links to standard devices[6]
25.	CKINIT, etc	117	Clock interrupt set up and handler[10]
26.	CICONV.PAL	246	Command interpreter[9]
27.	KS0011.PAL	296	CAMAC driver[11]
28.	VTINIT,etc	609	Subroutines used by PLOTA, mostly in overlay[2]
29.	ISORT.PAL	130	Calculates interger square roots
30.	BCDBIN.PAL	50	Converts BCD to binary
31.	PLOTA.PAL	1,351	Plots graphs on the memory scope (in overlay 2)[12]
32.	HCOPY.PAL	114	Transfers graph from the memory scope to the line printer (in overlay 2)[4]

TABLE XIII Con't

	FILE NAME	MEMORY WORDS	FUNCTION
33.	PLOTB.PAL	3,284	Plots graphs on the line printer (in overlay 3)
	D. DEC DOS Programs[3]		
34.	ODT	1,534	Program debugging, space used for general buffers unless ODT needed
35.	$LOAD,etc	292	Read in overlay segments
36.	-------	2,655	Resident monitor
37.	-------	1,353	"Monitor buffers", mostly device drivers
	E. Large Buffers of Data Storage (not included above)		
38.	BBCD	264	Constants and control for PAP, general constants
39.	TCDAT	512	Latest values, slow and fast data blocks
40.	PDP.MAC	512	Two general buffers (available at program start up)
41.	PAP.MAC	1,440	Input and output buffers for pulse data
42.	SDR.MAC	256	Status bytes for data in slow and fast blocks
43.	-------	$\leq$ 3,584	Up to 14 general buffers assigned at program start up

TABLE XIII Con't

TOTALS

A	16,207
B	1,165
C	7,567
D (Less ODT)	4,300
E	6,568
TOTAL --------	35,807

* TABLE XIV MEMORY USE FOR ONLINE PROGRAM WHEN RUNNING WITHOUT ODT (NUMBERS REFER TO ENTRIES IN TABLE XIII)

MEMORY WORDS	USE
2655	DOS monitor (36)
1353	Monitor buffers, mostly device drivers (37)
2345	BSX supervisor, including task table, links to drivers, and special drivers for CRT terminal and Decwriter (1,13,22,24)
4121	Overlay area. Display, disk block editing, mag tape initialization, misc. commands (0V1:17,18,19), (0V2:20,28,31,32), (0V3:21,33)
4486	Read in, analysis, logging, and real time display of pulse data (5,11,12,38,41)
5043	Read in, analysis and logging of fast and slow data, including device drivers: (2,3,7,9,10,15, 16,27,29,30,39,42)
591	Portion of list task in main segment, mostly to find and read in desired data from disk. (8)
1199	Control and formatting for line alphanumeric output devices (4, 23,28)
619	Command input and conversion (1,26)
556	Clock task and clock interrupt handler (1,25)

TABLE XIV Con't

MEMORY WORDS	USE
373	Overlay control and read in (14,35)
67	Disk I/O control (14)
233	Portion of mag tape task in main segment (6)
120	General buffer control (5)
4352	17 General buffers (12,40,43)
559	Unused memory
28672	Total (=28K)

FIGURE CAPTIONS

1. Schematic diagram of the liquid Helium flow to the superconducting magnet.

2. (a) Program steps for the read in of data during the bubble chamber pulse. (b) The computer beam track gates near beam time. In each case, the curve shows the pressure inside the chamber.

3. Pressure vs. volume diagram of one bubble chamber pulse. The area inside the curve is the work done on the chamber liquid done by the expansion system.

4. List of the gold chromel thermocouples during a chamber cooldown. Temperatures are given in degrees Kelvin. Note that both 15 minute and 60 minute rates are given; by comparing these two, the operator can tell whether the cooldown rate is increasing, decreasing, or remaining constant.

5. List of data concerning bubble chamber temperatures, cooling loops and expansion system during the heavy neon-hydrogen run. In this case, the mean of all readings between 0200-0400 on 24 May 1976 is given. The standard deviation (sigma) of each reading during the two hour period and the rate of change of the means with respect to the means for the previous two hour period are also given.

6. List of the data read into the slow data block every 1/16 hour.

7. List of the data read into the fast data block every 1/64 hour.

Figure Captions
-2-

8. List of the data read every chamber pulse.

9. Copy of a page display of pulse data on the CRT terminal screen. This page shows hadron beam timing using simulated data during a period when the bubble chamber was not running.

10. Typical operator commands needed to make a few variable summary, plot some of that data on the memory scope, make hard copy on the line printer from the memory scope, and to plot data directly on the line printer. The commands shown were actually used to make Figures 11, 12, 14 and 16.

11. Sample of a few variable summary list. Variables shown here are:

    1. Hydrogen storage dewar pressure
    2. Hydrogen storage dewar vent valve % open
    3. Neon-hydrogen storage dewar pressure
    4. Neon-hydrogen storage dewar condenser hydrogen level
    5. Neon-hydrogen storage dewar condenser supply valve % open
    6. Neon-hydrogen storage dewar dondenser vent valve % open

12. Graph of variable 1, Figure 11 vs. time. This graph was originally plotted on the memory scope with good resolution, but details were lost when it was hard copied on the line printer.

13. Graph of the liquid hydrogen storage dewar level vs. time for a two day period. Points are the means of all readings during a two hour interval and error bars are the standard deviations of the readings during that period. The figure was hard copied from the memory scope to the line printer.

Figure Captions
-3-

14. Four small graphs of variables vs. time, hard copied from the memory scope to the line printer. Clockwise, starting from the upper left, the variables are 1, 3, 6, 2 of Figure 11.

15. Scatter plot of one variable vs. another hard copied from the memory scope to the line printer. The x axis is the hydrogen storage dewar pressure and the y axis is the vent valve which relieves this pressure. The plot shows the action of the air controller which varies the vent valve to control dewar pressure. The data of Figures 11, 12 and 14 are included here, but a longer time interval, covering several control cycles, was chosen for the plot.

16. The same data as Figure 12, but plotted directly on the line printer.

17. The same data as Figure 13, but plotted directly on the line printer.

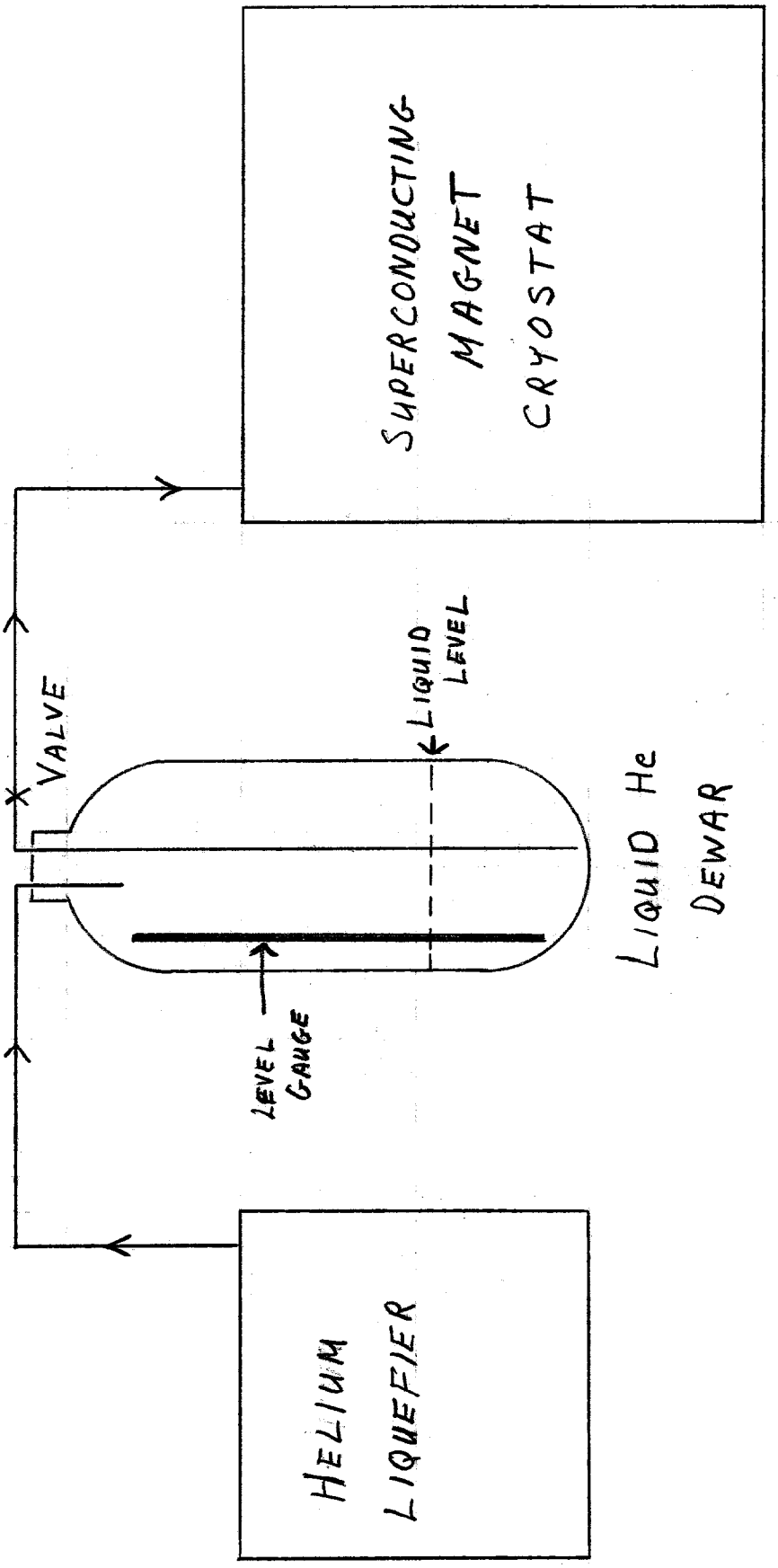

SUPERCONDUCTING
MAGNET
CRYOSTAT
VALVE
LIQUID
LEVEL
LEVEL
GAUGE
LIQUID He
DEWAR
HELIUM
LIQUEFIER

FIGURE 1.

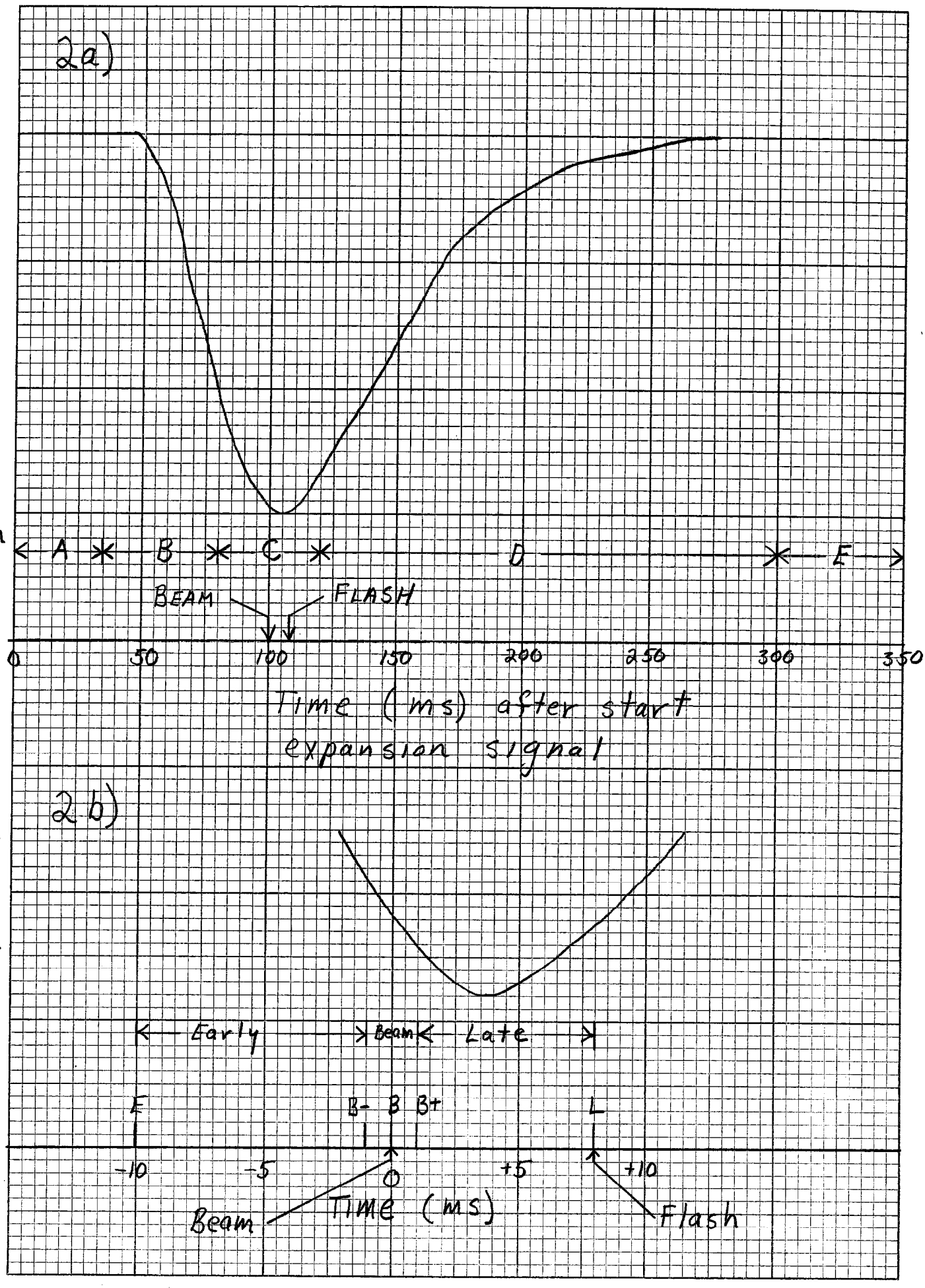


FIGURE 2.

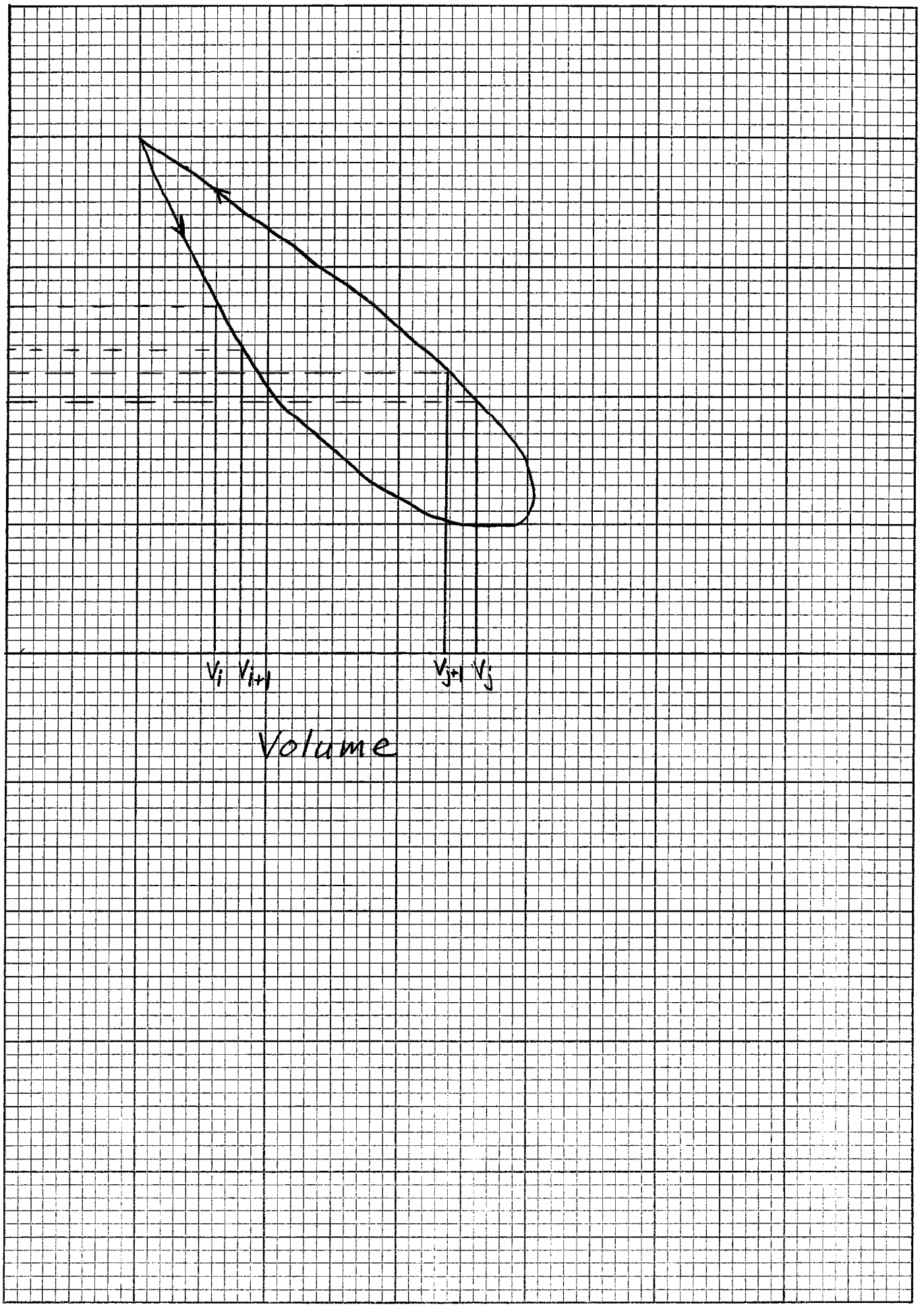


FIGURE 3

****** 15' BUBBLE CHAMBER ****** 06-APR-76 14:32:17

AT 14:30:00 06-APR-76 VALUES

TC	TEMP	15' R	60' R	TC	TEMP	15' R	60' R	TC	TEMP	15' R	60' R
100	88.5	-4.4	-5.0	119	81.3	-6.0	-5.0	138	112.6	-3.2	-3.7
101	103.2	-3.2	-3.9	120	*0	*0	*0	139	123.2	-2.4	-2.8
102	130.8	-2.8	-3.3	121	89.6	-4.0	-4.9	140	123.6	-2.4	-3.3
103	150.7	-2.8	6.2	122	88.7	-4.0	-4.9	141	109.9	-3.6	-3.6
104	*3	*0	*0	123	88.6	-4.0	-4.9	142	278.1	0.4	-0.4
105	111.0	-1.2	-1.5	124	89.0	-4.8	-5.0	143	107.2	-5.2	-5.2
106	114.0	-4.0	-4.2	125	*0	*0	*0	144	111.3	-8.4	-8.5
107	121.4	-3.6	-4.0	126	*0	*0	*0	160	*0	*0	*0
108	83.5	-4.8	-5.1	127	215.3	-0.8	-1.5	161	197.3	-0.4	-0.3
109	95.4	-3.6	-4.3	128	55.5	-1.2	-3.1	162	203.7	-1.6	-2.0
110	106.4	-3.6	-3.9	129	23.0	0.0	-1.1	163	219.7	-2.0	-2.8
111	106.5	-4.0	-4.1	130	106.9	-3.6	-4.0	164	276.9	-1.6	-2.2
112	*0	*0	*0	131	100.3	-4.0	-4.2	165	283.8	-1.2	-1.4
113	110.5	-3.2	-3.8	132	*0	*0	*0	166	285.8	-0.8	-1.3
114	106.1	-4.0	-4.1	133	*0	*0	*0	167	281.7	-0.8	-1.1
115	117.0	-3.2	-3.7	134	110.3	-3.6	-3.9	168	285.6	-0.8	-1.4
116	79.5	-4.8	-4.1	135	116.8	-2.8	-3.1	169	191.4	-4.8	-4.7
117	88.5	-4.4	-4.9	136	98.6	-3.6	-4.1	170	196.0	-6.8	-3.1
118	91.9	-3.2	-4.6	137	106.1	-3.6	-3.9	171	197.8	-5.6	-2.9
OPTICS NOZZEL				AVERAGE	116.2	-2.8	-3.4				
OPTICS INVAR				AVERAGE	110.8	-3.2	-3.6				
OPTICS LOOP OUTLET				AVERAGE	104.9	-4.0	-4.1				
CHAMBER BODY				AVERAGE	85.5	-4.8	-4.9				
PISTON EDGE				AVERAGE	109.2	-6.8	-6.9				
Z SECTION				AVERAGE	197.4	-2.4	-2.4				
PISTON SHAFT SEAL				AVERAGE	284.2	-0.8	-1.3				
NOZZEL MINUS INVAR				AVERAGE	5.4	0.4	0.2				

WHEN IDLE, DISPLAYING TC 166

FIGURE 4.

****** 15' BUBBLE CHAMBER ****** 25-MAY-76 10:39:26

AT 02:00:00 24-MAY-76 MEANS

CHAMBER TEMP IN PSIA

	TI	MEAN	SIGMA	2HR R	TI	MEAN	SIGMA	2HR R
CHAMBER AVG, TOP	341	101.52	5.02	-0.61	344	87.22	0.02	*0
BELOW D. JACKET	487	102.28	0.10	-0.13	484	101.03	0.05	-0.11
2' ABOVE EQUATOR	488	97.09	1.82	-0.30	485	98.94	0.04	-0.10
ABOVE PISTON	489	106.92	*0	0.02	486	106.04	*0	-0.02
PISTON SEALS	318	94.43	7.69	-3.61	319	100.78	0.94	-0.09
Z SECT. TOP, BOT	490	*0	*0	*0	491	*0	*0	*0
AVERAGE 484-489		99.62	2.54	-0.52	REF.	88.62	7.66	-3.57

EXPANSION SYSTEM IN PSIG

	PS	MEAN	SIGMA	2HR R	PS	MEAN	SIGMA	2HR R
DRIVE, BOUNCER	859	1114.5	3.6	-6.6	866	430.1	0.3	-0.3

CHAMBER LOOP VALVES % OPEN

BUBBLE, MAIN	168	2.5	4.0	-0.2	167	100.0	0.0	0.0
PUMP, PLENUM	163	100.0	0.0	0.0	164	100.0	0.0	0.0

DEVICE	UNITS	MEAN	SIGMA	2HR R	DEVICE	UNITS	MEAN	SIGMA	2HR R
MAIN LOOP									
PT 167	PSID	15.27	0.47	-0.05	PV 610	%	15.0	10.1	-0.3
PUMP LOOP									
TIC 163	SPEC	4.10	0.28	-0.02	PV 163	%	84.3	8.3	-1.3
DPT 613	PSID	13.25	0.08	0.09	FI 346	INCH	8.34	0.11	0.02
PLENUM LOOP									
PT 164	PSID	22.29	0.06	-0.02	PV 164	%	4.5	3.7	0.5
REFRIGERATION									
FI 205	INCH	0.36	2.42	0.33	PT 104	PSIG	27.01	0.37	-0.56
CHAMBER									
APV0 0	KJ	5.017	0.426	0.093	AB-P 0	MS	5.1	0.1	0.1
APV1 0	KJ	3.824	0.163	-0.127	ADP 0	PSI	46.28	0.11	-0.19
APV2 0	KJ	2.337	0.185	-0.169	AK1 0	INCH	1.91	0.01	-0.01
CPX 190	PSIA	119.30	0.03	-0.02					

FIGURE 5.

****** 15' BUBBLE CHAMBER ****** 04-OCT-76 14:37:25

SLOW DATA READ LIST. LAST MODIFIED AT 15:01:25 17-SEP-76 BLOCK 42

DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S	DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S
4	TC	100	DEGK	20647	400	4	25	TC	121	DEGK	20647	425	130
5	TC	101	DEGK	20647	401	10	26	TC	122	DEGK	20647	426	136
6	TC	102	DEGK	20647	402	16	27	TC	123	DEGK	20647	427	142
7	TC	103	DEGK	20647	403	22	28	TC	124	DEGK	20647	430	148
8	TC	104	DEGK	20647	404	28	29	TC	125	DEGK	20647	431	154
9	TC	105	DEGK	20647	405	34	30	TC	126	DEGK	20647	432	160
10	TC	106	DEGK	20647	406	40	31	TC	127	DEGK	20647	433	166
11	TC	107	DEGK	20647	407	46	32	TC	128	DEGK	20647	434	172
12	TC	108	DEGK	20647	410	52	33	TC	129	DEGK	20647	435	178
13	TC	109	DEGK	20647	411	58	34	TC	130	DEGK	20647	436	184
14	TC	110	DEGK	20647	412	64	35	TC	131	DEGK	20647	437	190
15	TC	111	DEGK	20647	413	70	36	TC	132	DEGK	20647	440	196
16	TC	112	DEGK	20647	414	76	37	TC	133	DEGK	20647	441	202
17	TC	113	DEGK	20647	415	82	38	TC	134	DEGK	20647	442	208
18	TC	114	DEGK	20647	416	88	39	TC	135	DEGK	20647	443	214
19	TC	115	DEGK	20647	417	94	40	TC	136	DEGK	20647	444	220
20	TC	116	DEGK	20647	420	100	41	TC	137	DEGK	20647	445	226
21	TC	117	DEGK	20647	421	106	42	TC	138	DEGK	20647	446	232
22	TC	118	DEGK	20647	422	112	43	TC	139	DEGK	20647	447	238
23	TC	119	DEGK	20647	423	118	44	TC	140	DEGK	20647	450	244
24	TC	120	DEGK	20647	424	124	45	TC	141	DEGK	20647	451	250

SLOW DATA READ LIST. LAST MODIFIED AT 21:32:25 19-AUG-76 BLOCK 43

DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S	DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S
46	TC	142	DEGK	20647	452	4	67	ZSA	5	DEGK	20647	0	130
47	TC	143	DEGK	20647	453	10	68	PSSA	6	DEGK	20647	0	136
48	TC	144	DEGK	20647	454	16	69	N-IA	7	DEGK	20647	0	142
49	TC	160	DEGK	20647	474	22	70	NONE	0	NONE	0	0	148
50	TC	161	DEGK	20647	475	28	71	NONE	0	NONE	0	0	154
51	TC	162	DEGK	20647	476	34	72	NONE	0	NONE	0	0	160
52	TC	163	DEGK	20647	477	40	73	NONE	0	NONE	0	0	166
53	TC	164	DEGK	20647	500	46	74	NONE	0	NONE	0	0	172
54	TC	165	DEGK	20647	501	52	75	NONE	0	NONE	0	0	178
55	TC	166	DEGK	20647	502	58	76	NONE	0	NONE	0	0	184
56	TC	167	DEGK	20647	503	64	77	NONE	0	NONE	0	0	190
57	TC	168	DEGK	20647	504	70	78	NONE	0	NONE	0	0	196
58	TC	169	DEGK	20647	505	76	79	NONE	0	NONE	0	0	202
59	TC	170	DEGK	20647	506	82	80	NONE	0	NONE	0	0	208
60	TC	171	DEGK	20647	507	88	81	NONE	0	NONE	0	0	214
61	NONE	0	NONE	0	0	94	82	NONE	0	NONE	0	0	220
62	ONA	0	DEGK	20647	0	100	83	NONE	0	NONE	0	0	226
63	OIA	1	DEGK	20647	0	106	84	NONE	0	NONE	0	0	232
64	OLOA	2	DEGK	20647	0	112	85	NONE	0	NONE	0	0	238
65	CBA	3	DEGK	20647	0	118	86	NONE	0	NONE	0	0	244
66	PEA	4	DEGK	20647	0	124	87	NONE	0	NONE	0	0	250

FIGURE 6a.

SLOW DATA READ LIST. LAST MODIFIED AT 13:02:14 03-SEP-76 BLOCK 44

DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S	DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S
88	TC	0	DEGK	20647	1000	4	109	TC	21	DEGK	20647	1025	130
89	TC	1	DEGK	20647	1001	10	110	TC	22	DEGK	20647	1026	136
90	TC	2	DEGK	20647	1002	16	111	TC	23	DEGK	20647	1027	142
91	TC	3	DEGK	20647	1003	22	112	TC	24	DEGK	20647	1030	148
92	TC	4	DEGK	20647	1004	28	113	TC	25	DEGK	20647	1031	154
93	TC	5	DEGK	20647	1005	34	114	TC	26	DEGK	20647	1032	160
94	TC	6	DEGK	20647	1006	40	115	TC	27	DEGK	20647	1033	166
95	TC	7	DEGK	20647	1007	46	116	TC	28	DEGK	20647	1034	172
96	TC	8	DEGK	20647	1010	52	117	TC	29	DEGK	20647	1035	178
97	TC	9	DEGK	20647	1011	58	118	TC	30	DEGK	20647	1036	184
98	TC	10	DEGK	20647	1012	64	119	TC	31	DEGK	20647	1037	190
99	TC	11	DEGK	20647	1013	70	120	TC	32	DEGK	20647	1040	196
100	TC	12	DEGK	20647	1014	76	121	TC	33	DEGK	20647	1041	202
101	TC	13	DEGK	20647	1015	82	122	TC	34	DEGK	20647	1042	208
102	TC	14	DEGK	20647	1016	88	123	TC	35	DEGK	20647	1043	214
103	TC	15	DEGK	20647	1017	94	124	TC	36	DEGK	20647	1044	220
104	TC	16	DEGK	20647	1020	100	125	TC	37	DEGK	20647	1045	226
105	TC	17	DEGK	20647	1021	106	126	TC	38	DEGK	20647	1046	232
106	TC	18	DEGK	20647	1022	112	127	TC	39	DEGK	20647	1047	238
107	TC	19	DEGK	20647	1023	118	128	TC	40	DEGK	20647	1050	244
108	TC	20	DEGK	20647	1024	124	129	TC	41	DEGK	20647	1051	250

SLOW DATA READ LIST. LAST MODIFIED AT 15:01:12 17-SEP-76 BLOCK 45

DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S	DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S
130	TC	42	DEGK	20647	1052	4	151	TC	81	DEGK	20647	1121	130
131	TC	43	DEGK	20647	1053	10	152	TC	82	DEGK	20647	1122	136
132	TC	45	DEGK	20647	1055	16	153	TC	83	DEGK	20647	1123	142
133	TC	46	DEGK	20647	1056	22	154	TC	84	DEGK	20647	1124	148
134	TC	47	DEGK	20647	1057	28	155	TC	85	DEGK	20647	1125	154
135	TC	48	DEGK	20647	1060	34	156	TC	86	DEGK	20647	1126	160
136	TC	49	DEGK	20647	1061	40	157	TC	87	DEGK	20647	1127	166
137	TC	50	DEGK	20647	1062	46	158	TC	88	DEGK	20647	1130	172
138	TC	51	DEGK	20647	1063	52	159	TC	89	DEGK	20647	1131	178
139	TC	55	DEGK	20647	1067	58	160	TC	90	DEGK	20647	1132	184
140	TC	56	DEGK	20647	1070	64	161	TC	91	DEGK	20647	1133	190
141	TC	57	DEGK	20647	1071	70	162	TC	92	DEGK	20647	1134	196
142	TC	58	DEGK	20647	1072	76	163	TC	93	DEGK	20647	1135	202
143	TC	59	DEGK	20647	1073	82	164	TC	94	DEGK	20647	1136	208
144	TC	60	DEGK	20647	1074	88	165	TC	95	DEGK	20647	1137	214
145	TC	61	DEGK	20647	1075	94	166	TC	96	DEGK	20647	1140	220
146	TC	62	DEGK	20647	1076	100	167	NONE	0	NONE	0	0	226
147	TC	70	DEGK	20647	1106	106	168	NONE	0	NONE	0	0	232
148	TC	71	DEGK	20647	1107	112	169	NONE	0	NONE	0	0	238
149	TC	79	DEGK	20647	1117	118	170	NONE	0	NONE	0	0	244
150	TC	80	DEGK	20647	1120	124	171	NONE	0	NONE	0	0	250

FIGURE 6b.

SLOW DATA READ LIST. LAST MODIFIED AT 15:42:53 16-AUG-76 BLOCK 46

DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S	DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S
172	NONE	0	NONE	0	0	4	193	NONE	0	NONE	0	0	130
173	NONE	0	NONE	0	0	10	194	NONE	0	NONE	0	0	136
174	NONE	0	NONE	0	0	16	195	NONE	0	NONE	0	0	142
175	NONE	0	NONE	0	0	22	196	NONE	0	NONE	0	0	148
176	NONE	0	NONE	0	0	28	197	NONE	0	NONE	0	0	154
177	NONE	0	NONE	0	0	34	198	NONE	0	NONE	0	0	160
178	NONE	0	NONE	0	0	40	199	NONE	0	NONE	0	0	166
179	NONE	0	NONE	0	0	46	200	NONE	0	NONE	0	0	172
180	NONE	0	NONE	0	0	52	201	NONE	0	NONE	0	0	178
181	NONE	0	NONE	0	0	58	202	NONE	0	NONE	0	0	184
182	NONE	0	NONE	0	0	64	203	NONE	0	NONE	0	0	190
183	NONE	0	NONE	0	0	70	204	NONE	0	NONE	0	0	196
184	NONE	0	NONE	0	0	76	205	NONE	0	NONE	0	0	202
185	NONE	0	NONE	0	0	82	206	NONE	0	NONE	0	0	208
186	NONE	0	NONE	0	0	88	207	NONE	0	NONE	0	0	214
187	NONE	0	NONE	0	0	94	208	NONE	0	NONE	0	0	220
188	NONE	0	NONE	0	0	100	209	NONE	0	NONE	0	0	226
189	NONE	0	NONE	0	0	106	210	NONE	0	NONE	0	0	232
190	NONE	0	NONE	0	0	112	211	NONE	0	NONE	0	0	238
191	NONE	0	NONE	0	0	118	212	NONE	0	NONE	0	0	244
192	NONE	0	NONE	0	0	124	213	NONE	0	NONE	0	0	250

SLOW DATA READ LIST. LAST MODIFIED AT 13:03:02 03-SEP-76 BLOCK 47

DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S	DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S
214	NONE	0	NONE	0	0	4	235	NONE	0	NONE	0	0	130
215	NONE	0	NONE	0	0	10	236	NONE	0	NONE	0	0	136
216	NONE	0	NONE	0	0	16	237	NONE	0	NONE	0	0	142
217	NONE	0	NONE	0	0	22	238	NONE	0	NONE	0	0	148
218	NONE	0	NONE	0	0	28	239	NONE	0	NONE	0	0	154
219	NONE	0	NONE	0	0	34	240	APV0	0	KJ	60647	0	160
220	NONE	0	NONE	0	0	40	241	APV1	0	KJ	60647	0	166
221	NONE	0	NONE	0	0	46	242	APV2	0	KJ	60647	0	172
222	NONE	0	NONE	0	0	52	243	AB-P	0	MS	20647	0	178
223	NONE	0	NONE	0	0	58	244	ADP	0	PSI	40647	0	184
224	NONE	0	NONE	0	0	64	245	AK1	0	INCH	40647	0	190
225	NONE	0	NONE	0	0	70	246	APV	168	%	20647	0	196
226	NONE	0	NONE	0	0	76	247	APV	167	%	20647	0	202
227	NONE	0	NONE	0	0	82	248	APV	163	%	20647	0	208
228	NONE	0	NONE	0	0	88	249	APV	164	%	20647	0	214
229	NONE	0	NONE	0	0	94	250	APV	270	%	20647	0	220
230	NONE	0	NONE	0	0	100	251	APV	327	%	20647	0	226
231	NONE	0	NONE	0	0	106	252	NONE	0	NONE	0	0	232
232	NONE	0	NONE	0	0	112	253	NONE	0	NONE	0	0	238
233	NONE	0	NONE	0	0	118	254	NONE	0	NONE	0	0	244
234	NONE	0	NONE	0	0	124	255	NONE	0	NONE	0	0	250

FIGURE 6C.

****** 15' BUBBLE CHAMBER ****** 04-OCT-76 14:38:04

FAST DATA READ LIST. LAST MODIFIED AT 14:43:46 16-AUG-76 BLOCK 72

DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S	DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S
4	NONE	0	NONE	0	0	4	25	NONE	0	NONE	0	0	130
5	NONE	0	NONE	0	0	10	26	NONE	0	NONE	0	0	136
6	NONE	0	NONE	0	0	16	27	NONE	0	NONE	0	0	142
7	NONE	0	NONE	0	0	22	28	NONE	0	NONE	0	0	148
8	NONE	0	NONE	0	0	28	29	NONE	0	NONE	0	0	154
9	NONE	0	NONE	0	0	34	30	NONE	0	NONE	0	0	160
10	NONE	0	NONE	0	0	40	31	NONE	0	NONE	0	0	166
11	NONE	0	NONE	0	0	46	32	NONE	0	NONE	0	0	172
12	NONE	0	NONE	0	0	52	33	NONE	0	NONE	0	0	178
13	NONE	0	NONE	0	0	58	34	NONE	0	NONE	0	0	184
14	NONE	0	NONE	0	0	64	35	NONE	0	NONE	0	0	190
15	NONE	0	NONE	0	0	70	36	NONE	0	NONE	0	0	196
16	NONE	0	NONE	0	0	76	37	NONE	0	NONE	0	0	202
17	NONE	0	NONE	0	0	82	38	NONE	0	NONE	0	0	208
18	NONE	0	NONE	0	0	88	39	NONE	0	NONE	0	0	214
19	NONE	0	NONE	0	0	94	40	NONE	0	NONE	0	0	220
20	NONE	0	NONE	0	0	100	41	NONE	0	NONE	0	0	226
21	NONE	0	NONE	0	0	106	42	NONE	0	NONE	0	0	232
22	NONE	0	NONE	0	0	112	43	NONE	0	NONE	0	0	238
23	NONE	0	NONE	0	0	118	44	NONE	0	NONE	0	0	244
24	NONE	0	NONE	0	0	124	45	NONE	0	NONE	0	0	250

FAST DATA READ LIST. LAST MODIFIED AT 17:10:50 12-AUG-76 BLOCK 73

DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S	DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S
46	NONE	0	NONE	0	0	4	67	NONE	0	NONE	0	0	130
47	NONE	0	NONE	0	0	10	68	NONE	0	NONE	0	0	136
48	NONE	0	NONE	0	0	16	69	NONE	0	NONE	0	0	142
49	NONE	0	NONE	0	0	22	70	NONE	0	NONE	0	0	148
50	NONE	0	NONE	0	0	28	71	NONE	0	NONE	0	0	154
51	NONE	0	NONE	0	0	34	72	NONE	0	NONE	0	0	160
52	NONE	0	NONE	0	0	40	73	NONE	0	NONE	0	0	166
53	NONE	0	NONE	0	0	46	74	NONE	0	NONE	0	0	172
54	NONE	0	NONE	0	0	52	75	NONE	0	NONE	0	0	178
55	NONE	0	NONE	0	0	58	76	NONE	0	NONE	0	0	184
56	NONE	0	NONE	0	0	64	77	NONE	0	NONE	0	0	190
57	NONE	0	NONE	0	0	70	78	NONE	0	NONE	0	0	196
58	NONE	0	NONE	0	0	76	79	NONE	0	NONE	0	0	202
59	NONE	0	NONE	0	0	82	80	NONE	0	NONE	0	0	208
60	NONE	0	NONE	0	0	88	81	NONE	0	NONE	0	0	214
61	NONE	0	NONE	0	0	94	82	NONE	0	NONE	0	0	220
62	NONE	0	NONE	0	0	100	83	NONE	0	NONE	0	0	226
63	NONE	0	NONE	0	0	106	84	NONE	0	NONE	0	0	232
64	NONE	0	NONE	0	0	112	85	NONE	0	NONE	0	0	238
65	NONE	0	NONE	0	0	118	86	NONE	0	NONE	0	0	244
66	NONE	0	NONE	0	0	124	87	NONE	0	NONE	0	0	250

FIGURE 7a.

FAST DATA READ LIST. LAST MODIFIED AT 14:54:22 17-SEP-76 BLOCK 74

DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S	DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S
88	NONE	0	NONE	0	0	4	109	NONE	0	NONE	0	0	130
89	NONE	0	NONE	0	0	10	110	NONE	0	NONE	0	0	136
90	NONE	0	NONE	0	0	16	111	NONE	0	NONE	0	0	142
91	NONE	0	NONE	0	0	22	112	NONE	0	NONE	0	0	148
92	NONE	0	NONE	0	0	28	113	NONE	0	NONE	0	0	154
93	NONE	0	NONE	0	0	34	114	NONE	0	NONE	0	0	160
94	NONE	0	NONE	0	0	40	115	NONE	0	NONE	0	0	166
95	NONE	0	NONE	0	0	46	116	NONE	0	NONE	0	0	172
96	NONE	0	NONE	0	0	52	117	NONE	0	NONE	0	0	178
97	NONE	0	NONE	0	0	58	118	NONE	0	NONE	0	0	184
98	NONE	0	NONE	0	0	64	119	NONE	0	NONE	0	0	190
99	NONE	0	NONE	0	0	70	120	NONE	0	NONE	0	0	196
100	NONE	0	NONE	0	0	76	121	NONE	0	NONE	0	0	202
101	NONE	0	NONE	0	0	82	122	NONE	0	NONE	0	0	208
102	NONE	0	NONE	0	0	88	123	NONE	0	NONE	0	0	214
103	NONE	0	NONE	0	0	94	124	NONE	0	NONE	0	0	220
104	NONE	0	NONE	0	0	100	125	NONE	0	NONE	0	0	226
105	NONE	0	NONE	0	0	106	126	NONE	0	NONE	0	0	232
106	NONE	0	NONE	0	0	112	127	NONE	0	NONE	0	0	238
107	NONE	0	NONE	0	0	118	128	NONE	0	NONE	0	0	244
108	NONE	0	NONE	0	0	124	129	NONE	0	NONE	0	0	250

FAST DATA READ LIST. LAST MODIFIED AT 21:09:06 21-SEP-76 BLOCK 75

DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S	DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S
130	TI	341	PSID	40647	3000	4	151	PVA	150	%	20647	3400	130
131	TI	490	PSID	40647	3400	10	152	PT	226	PSIG	647	3400	136
132	PT	161	PSIG	20647	3400	16	153	PVB	104	%	20647	3400	142
133	PV	610	%	20647	3400	22	154	TI	489	PSID	40647	3400	148
134	DPI	130	INCH	40647	3400	28	155	TI	486	PSID	40647	3400	154
135	PT	150	PSIG	20647	3400	34	156	PV	336	%	20647	3400	160
136	LI	224	INCH	40647	3400	40	157	PV	228	%	20647	3400	166
137	PT	104	PSIG	40647	3400	46	158	PT	140	PSIG	20647	3400	172
138	TI	487	PSID	40647	3400	52	159	PVB	150	%	20647	3400	178
139	TI	485	PSID	40647	3400	58	160	PV	226	%	20647	3400	184
140	PT	164	PSID	40647	3400	64	161	LI	129	INCH	40647	3400	190
141	DPT	336	INCH	40647	3400	70	162	TI	484	PSID	40647	3400	196
142	DPT	134	INCH	40647	3400	76	163	PT	184	PSIA	20647	3400	202
143	DPT	151	INCH	40647	3400	82	164	PV	159	%	20647	3400	208
144	PV	224	%	20647	3400	88	165	PT	232	PSIG	647	3400	214
145	PVA	104	%	20647	3400	94	166	PVA	140	%	20647	3400	220
146	TI	488	PSID	40647	3400	100	167	DPT	147	INCH	40647	3400	226
147	TI	318	PSID	40647	3400	106	168	PV	227	%	20647	3400	232
148	PT	167	PSID	40647	3400	112	169	PV	105	%	20647	3400	238
149	TIC	163	SPEC	40647	3400	118	170	TI	344	PSID	40647	3400	244
150	PV	134	%	20647	3400	124	171	PV	183	%	20647	3400	250

FIGURE 7b.

FAST DATA READ LIST. LAST MODIFIED AT 13:08:39 14-AUG-76 BLOCK 76

DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S	DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S
172	PV	161	%	20647	3400	4	193	PV	111	%	20647	3400	130
173	PT	101	PSIG	40647	3400	10	194	VPTR	0	PSIA	40647	1414	136
174	PVB	140	%	20647	3400	16	195	A/DR	0	PSIA	40647	2436	142
175	PV	147	%	20647	3400	22	196	RELV	0	PSIA	40647	0	148
176	FI	205	INCH	40647	3400	28	197	LI Z	455	CM D	40647	1416	154
177	PT	114	PSIG	20647	3400	34	198	LI Z	455	L D	20647	1416	160
178	TI	319	PSID	40647	3400	40	199	NEON	0	%	20647	1715	166
179	FI	346	INCH	40647	3400	46	200	TI	341	PSIA	40647	0	172
180	PV	163	%	20647	3400	52	201	TI	344	PSIA	40647	0	178
181	PV	101	%	20647	3400	58	202	TI	487	PSIA	40647	0	184
182	PV	139	%	20647	3400	64	203	TI	484	PSIA	40647	0	190
183	PV	155	%	20647	3400	70	204	TI	488	PSIA	40647	0	196
184	LI U	8002	INCH	20647	3400	76	205	TI	485	PSIA	40647	0	202
185	PT	111	PSIG	20647	3400	82	206	TI	489	PSIA	40647	0	208
186	TI	491	PSID	40647	3400	88	207	TI	486	PSIA	40647	0	214
187	DPT	613	PSID	40647	3400	94	208	TI	318	PSIA	40647	0	220
188	PV	164	%	20647	3400	100	209	TI	319	PSIA	40647	0	226
189	FI	103	INCH	20647	3400	106	210	TI	490	PSIA	40647	0	232
190	DPT	110	INCH	40647	3400	112	211	TI	491	PSIA	40647	0	238
191	PTIA	8000	PSIG	20647	3400	118	212	CTIA	0	PSIA	40647	0	244
192	PTEA	8001	PSIG	647	3400	124	213	NONE	0	NONE	0	0	250

FAST DATA READ LIST. LAST MODIFIED AT 17:43:53 17-SEP-76 BLOCK 77

DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S	DP#	NAME	NO.	UNIT	FRMT	D/SA	O/S
214	RA	0	ROLL	647	4001	4	235	RA/D	5	CHN#	40647	2405	130
215	FA	0	FRAM	647	4002	10	236	RA/D	6	CHN#	40647	2406	136
216	RB	0	ROLL	647	4003	16	237	RA/D	7	CHN#	40647	2407	142
217	FB	0	FRAM	647	4004	22	238	PS	859	PSIG	20647	1670	148
218	MAG	0	AMP	647	4005	28	239	PS	866	PSIG	20647	1471	154
219	MAG	1	VOLT	20647	4006	34	240	NONE	0	NONE	0	0	160
220	M PS	2	VOLT	40647	4007	40	241	NONE	0	NONE	0	0	166
221	CPX	190	PSIA	40647	4010	46	242	NONE	0	NONE	0	0	172
222	A/D	0	PSIA	40647	2400	52	243	NONE	0	NONE	0	0	178
223	A/D	1	PSIA	40647	2401	58	244	NONE	0	NONE	0	0	184
224	A/D	2	PSIA	40647	2402	64	245	NONE	0	NONE	0	0	190
225	A/D	3	PSIG	20647	2403	70	246	NONE	0	NONE	0	0	196
226	A/D	4	PSIA	40647	2404	76	247	NONE	0	NONE	0	0	202
227	A/D	5	PSIA	40647	2405	82	248	NONE	0	NONE	0	0	208
228	A/D	6	PSIA	40647	2406	88	249	NONE	0	NONE	0	0	214
229	A/D	7	PSIA	40647	2407	94	250	NONE	0	NONE	0	0	220
230	RA/D	0	CHN#	40647	2400	100	251	NONE	0	NONE	0	0	226
231	RA/D	1	CHN#	40647	2401	106	252	NONE	0	NONE	0	0	232
232	RA/D	2	CHN#	40647	2402	112	253	NONE	0	NONE	0	0	238
233	RA/D	3	CHN#	40647	2403	118	254	NONE	0	NONE	0	0	244
234	RA/D	4	CHN#	40647	2404	124	255	NONE	0	NONE	0	0	250

FIGURE 7C.

****** 15' BUBBLE CHAMBER ****** 04-OCT-76 14:58:17

PULSE DATA READ LIST. LAST MODIFIED AT 10:23:36 13-APR-76 BLOCK 101

DP#	NAME AND UNIT	FRMT	O/S	DP#	NAME AND UNIT	FRMT	O/S
4	EXP A ROLL	447	4	19	ARC 1 KV	60647	124
5	EXP A FRAME	447	12	20	ARC 2 KV	60647	132
6	EXP B ROLL	447	20	21	ARC 3 KV	60647	140
7	EXP B FRAME	447	28	22	ARC 4 KV	60647	148
8	MAGNET I AMP	647	36	23	ARC 5 KV	60647	156
9	CP PSIA	40647	44	24	ARC 6 KV	60647	164
10	VP PSIA	40647	52	25	A/D0 STAT PSIA	40647	172
11	TIME PP SEC	60647	60	26	A/D1 STAT PSIA	40647	180
12	PULSE NUMBER	447	68	27	A/D2 STAT PSIA	40647	188
13	SCR MS	20647	76	28	A/D3 STAT PSIG	20647	196
14	ZCR MS	20647	84	29	A/D4 STAT PSIA	40647	204
15	EARLY GATE MS	20647	92	30	A/D5 STAT PSIA	40647	212
16	BEAM GATE MS	20647	100	31	A/D6 STAT INCH	40647	220
17	LATE GATE MS	20647	108	32	A/D7 STAT INCH	40647	228
18	DD MS	20647	116	33	SPARE	0	236
				34	SPARE	0	244

PULSE DATA READ LIST. LAST MODIFIED AT 22:18:25 16-APR-76 BLOCK 102

DP#	NAME AND UNIT	FRMT	O/S	DP#	NAME AND UNIT	FRMT	O/S
35	SPARE	0	4	50	S2 NS COUNTS	40647	124
36	SPARE	0	12	51	S3 NS COUNTS	40647	132
37	SPARE	0	20	52	S2 EARLY CNTS	40647	140
38	SPARE	0	28	53	S3 EARLY CNTS	40647	148
39	LATCH CLOSE MS	20647	36	54	S2 B-TO B CNTS	40647	156
40	RECOM CLOSE MS	20647	44	55	S3 B-TO B CNTS	40647	164
41	EXP CLOSED MS	20647	52	56	S2 BEAM COUNTS	40647	172
42	EXP A BEAM MS	20647	60	57	S3 BEAM COUNTS	40647	180
43	EXP B BEAM MS	20647	68	58	S2 LATE COUNTS	40647	188
44	ARC TIME MS	20647	76	59	S3 LATE COUNTS	40647	196
45	DATA BOX MS	20647	84	60	S0EC	0	204
46	CAMERA TIME MS	20647	92	61	S1EC	0	212
47	LATCH OPEN MS	20647	100	62	S0B-C	0	220
48	RECOM OPEN MS	20647	108	63	S1B-C	0	228
49	PROG BEAM MS	20647	116	64	S0B+C	0	236
				65	S1B+C	0	244

PULSE DATA READ LIST. LAST MODIFIED AT 10:25:37 13-APR-76 BLOCK 103

DP#	NAME AND UNIT	FRMT	O/S	DP#	NAME AND UNIT	FRMT	O/S
66	S0LC	60647	4	81	A/D2 EXB PSIA	40647	124
67	S1LC	60647	12	82	A/D2 B TIME MS	20647	132
68	AREA PV0 KJ	60647	20	83	A/D3 EXA PSIG	20647	140
69	AREA PV1 KJ	60647	28	84	A/D3 A TIME MS	20647	148
70	AREA PV2 KJ	60647	36	85	A/D3 EXB PSIG	20647	156
71	A/D0 EXA PSIA	40647	44	86	A/D3 B TIME MS	20647	164
72	A/D0 A TIME MS	20647	52	87	A/D4 EXA PSIA	40647	172
73	A/D0 EXB PSIA	40647	60	88	A/D4 A TIME MS	20647	180
74	A/D0 B TIME MS	20647	68	89	A/D4 EXB PSIA	40647	188
75	A/D1 EXA PSIA	40647	76	90	A/D4 B TIME MS	20647	196
76	A/D1 A TIME MS	20647	84	91	A/D5 EXA PSIA	40647	204
77	A/D1 EXB PSIA	40647	92	92	A/D5 A TIME MS	20647	212
78	A/D1 B TIME MS	20647	100	93	A/D5 EXB PSIA	40647	220
79	A/D2 EXA PSIA	40647	108	94	A/D5 B TIME MS	20647	228
80	A/D2 A TIME MS	20647	116	95	A/D6 EXA INCH	40647	236
				96	A/D6 A TIME MS	20647	244

FIGURE 8a.

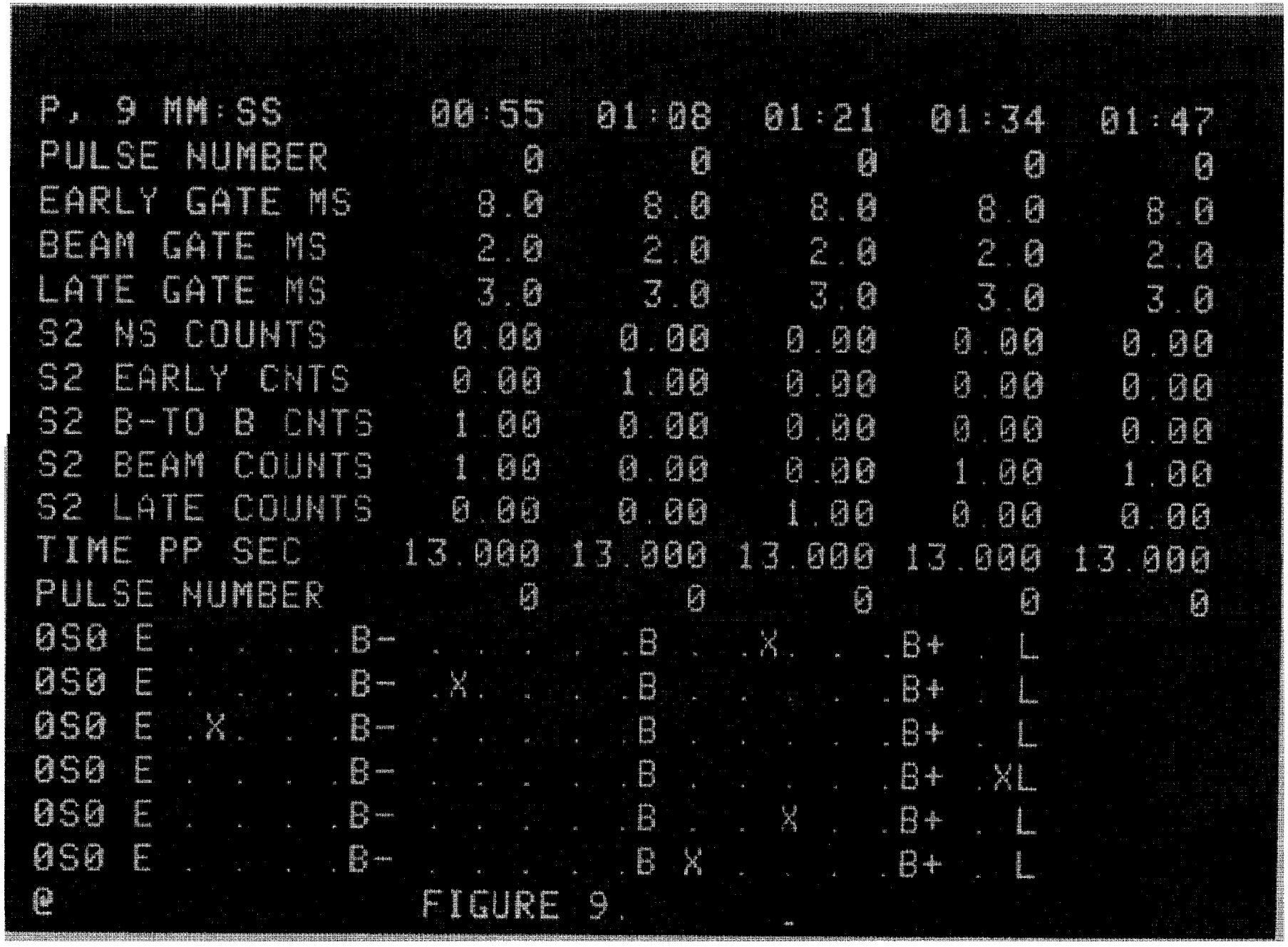
PULSE DATA READ LIST. LAST MODIFIED AT 12:17:31 01-SEP-76 BLOCK 104

DP#	NAME AND UNIT	FRMT	O/S	DP#	NAME AND UNIT	FRMT	O/S
97	A/D6 EXB INCH	40647	4	112	SPARE	0	124
98	A/D6 B TIME MS	20647	12	113	SPARE	0	132
99	A/D7 EXA INCH	40647	20	114	SPARE	0	140
100	A/D7 A TIME MS	20647	28	115	SPARE	0	148
101	A/D7 EXB INCH	40647	36	116	SPARE	0	156
102	A/D7 B TIME MS	20647	44	117	SPARE	0	164
103	PMIN TO B MS	20647	52	118	SPARE	0	172
104	BEAM TO ARC MS	20647	60	119	SPARE	0	180
105	A EFFICIENCY %	40647	68	120	SPARE	0	188
106	B EFFICIENCY %	40647	76	121	SPARE	0	196
107	DELTA P PSI	40647	84	122	SPARE	0	204
108	STROKE INCH	40647	92	123	# ACC CLOCK T1	647	212
109	LCTR1	647	100	124	# BEAM IN MR	647	220
110	LCTR5	647	108	125	# BEAM IN NL	647	228
111	SPARE	0	116	126	SPARE SCALER	647	236
				127	PULSES A EXP	647	244

PULSE DATA READ LIST. LAST MODIFIED AT 15:48:43 01-SEP-76 BLOCK 105

DP#	NAME AND UNIT	FRMT	O/S
128	PULSES B EXP	647	4

FIGURE 8b.



```
P, 9 MM:SS          00:55   01:08   01:21   01:34   01:47
PULSE NUMBER            0       0       0       0       0
EARLY GATE MS         8.0     8.0     8.0     8.0     8.0
BEAM GATE MS          2.0     2.0     2.0     2.0     2.0
LATE GATE MS          3.0     3.0     3.0     3.0     3.0
S2 NS COUNTS         0.00    0.00    0.00    0.00    0.00
S2 EARLY CNTS        0.00    1.00    0.00    0.00    0.00
S2 B-TO B CNTS       1.00    0.00    0.00    0.00    0.00
S2 BEAM COUNTS       1.00    0.00    0.00    1.00    1.00
S2 LATE COUNTS       0.00    0.00    1.00    0.00    0.00
TIME PP SEC        13.000  13.000  13.000  13.000  13.000
PULSE NUMBER            0       0       0       0       0
0S0 E . . . . .B- . . . . . .B . . .X. . . .B+ . . L
0S0 E . . . . .B- . .X. . . . .B . . . . . .B+ . . L
0S0 E . .X. . . .B- . . . . . .B . . . . . .B+ . . L
0S0 E . . . . .B- . . . . . .B . . . . . .B+ . .XL
0S0 E . . . . .B- . . . . . .B . . . X . . .B+ . . L
0S0 E . . . . .B- . . . . . .B .X. . . . .B+ . . L
@                    FIGURE 9.
```

TM-687

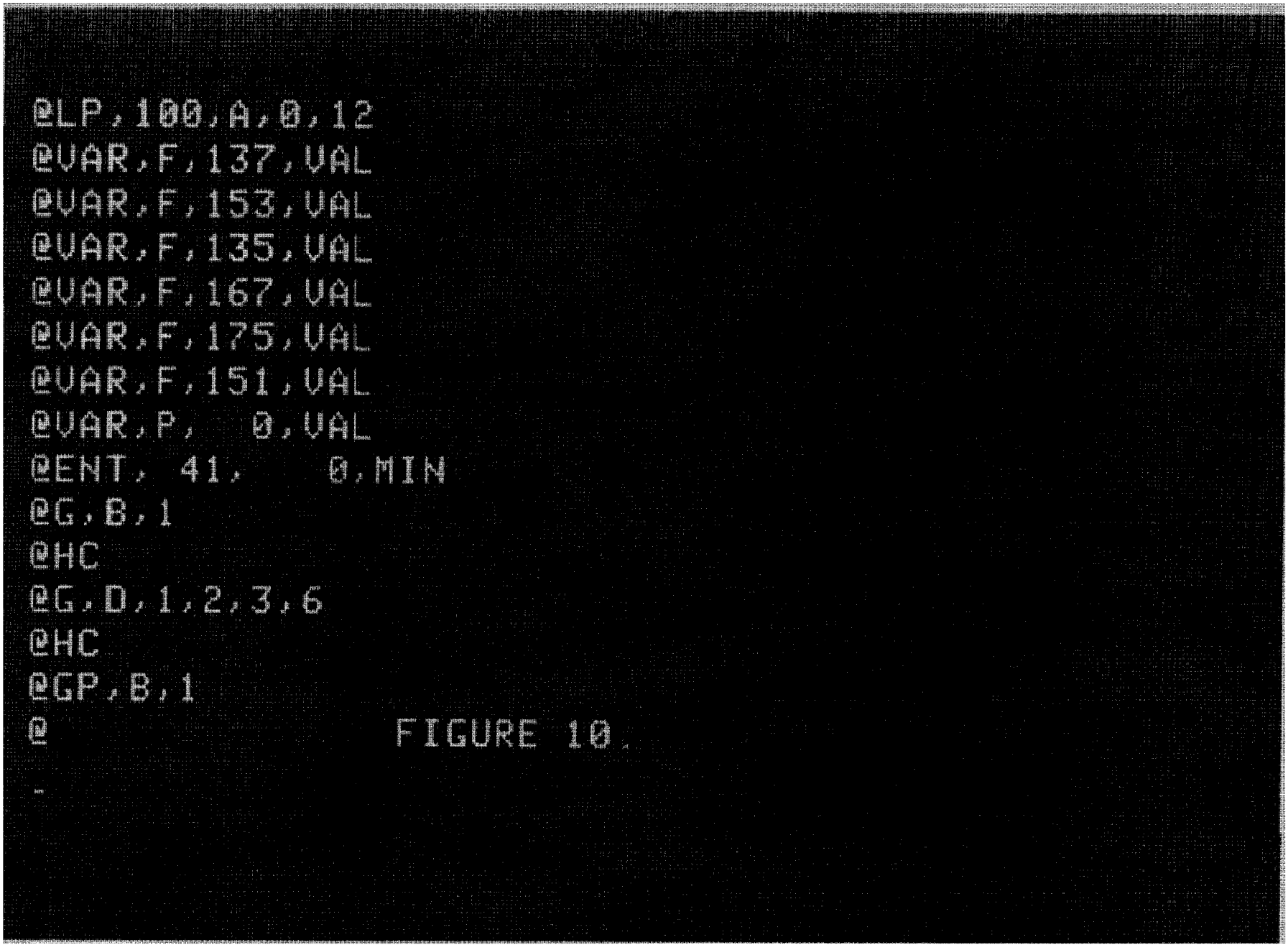

@LP,100,A,0,12
@VAR,F,137,VAL
@VAR,F,153,VAL
@VAR,F,135,VAL
@VAR,F,167,VAL
@VAR,F,175,VAL
@VAR,F,151,VAL
@VAR,P, 0,VAL
@ENT, 41, 0,MIN
@G,B,1
@HC
@G,D,1,2,3,6
@HC
@GP,B,1
@

FIGURE 10.

****** 15' BUBBLE CHAMBER ****** 04-OCT-76 14:09:42

1. PT 104 PSIG VALUE (F,137)
2. PVB 104 % VALUE (F,153)
3. PT 150 PSIG VALUE (F,135)
4. DPT 147 INCH VALUE (F,167)
5. PV 147 % VALUE (F,175)
6. PVA 150 % VALUE (F,151)

			1	2	3	4	5	6	7
04-OCT-76	11:22:30	-40	32.38	0.0	36.3	2.59	0.0	8.2	
04-OCT-76	11:23:26	-39	32.41	0.0	36.4	2.60	0.0	11.8	
04-OCT-76	11:24:22	-38	32.38	0.0	36.5	2.64	0.0	15.0	
04-OCT-76	11:25:19	-37	32.45	0.0	36.4	3.25	0.0	21.5	
04-OCT-76	11:26:15	-36	32.41	0.0	36.3	2.24	34.5	15.5	
04-OCT-76	11:27:11	-35	32.41	0.0	36.3	2.56	0.0	6.3	
04-OCT-76	11:28:07	-34	32.45	0.0	36.3	2.62	0.0	6.1	
04-OCT-76	11:29:04	-33	32.45	0.0	36.4	2.64	0.0	8.1	
04-OCT-76	11:30:00	-32	32.51	0.0	36.5	2.64	0.0	12.9	
04-OCT-76	11:30:56	-31	32.48	0.0	36.5	2.57	0.0	19.9	
04-OCT-76	11:31:52	-30	32.48	0.0	36.4	4.50	0.0	17.0	
04-OCT-76	11:32:49	-29	32.48	0.0	36.3	2.49	25.0	8.8	
04-OCT-76	11:33:45	-28	32.45	0.0	36.3	2.53	23.9	9.1	
04-OCT-76	11:34:41	-27	32.48	0.0	36.4	2.52	27.4	12.4	
04-OCT-76	11:35:37	-26	32.48	0.0	36.4	2.53	30.5	16.7	
04-OCT-76	11:36:34	-25	32.45	0.4	36.5	4.05	0.0	21.0	
04-OCT-76	11:37:30	-24	32.48	1.5	36.3	2.52	7.6	11.8	
04-OCT-76	11:38:26	-23	32.48	2.8	36.3	2.59	0.0	5.7	
04-OCT-76	11:39:22	-22	32.48	4.2	36.4	2.60	0.0	7.2	
04-OCT-76	11:40:19	-21	32.38	4.7	36.4	2.62	0.0	11.0	
04-OCT-76	11:41:15	-20	32.26	4.2	36.5	2.63	0.0	15.0	
04-OCT-76	11:42:11	-19	32.13	1.9	36.5	2.53	0.0	19.5	
04-OCT-76	11:43:07	-18	32.13	0.0	36.4	3.82	0.0	16.8	
04-OCT-76	11:44:04	-17	32.13	0.0	36.3	2.62	0.0	7.6	
04-OCT-76	11:45:00	-16	32.20	0.0	36.4	2.63	0.0	7.2	
04-OCT-76	11:45:56	-15	32.20	0.0	36.3	2.63	0.0	10.4	
04-OCT-76	11:46:52	-14	32.16	0.0	36.4	2.68	0.0	14.2	
04-OCT-76	11:47:49	-13	32.20	0.0	36.4	2.60	0.0	20.0	
04-OCT-76	11:48:45	-12	32.20	0.0	36.3	4.07	0.0	17.9	
04-OCT-76	11:49:41	-11	32.23	0.0	36.3	2.57	1.3	5.6	
04-OCT-76	11:50:37	-10	32.26	0.0	36.3	2.62	0.0	5.7	
04-OCT-76	11:51:34	-9	32.26	0.0	36.3	2.61	0.0	8.5	
04-OCT-76	11:52:30	-8	32.32	0.0	36.4	2.64	0.0	12.4	
04-OCT-76	11:53:26	-7	32.29	0.0	36.5	2.64	0.0	17.5	
04-OCT-76	11:54:22	-6	32.32	0.0	36.5	5.71	0.0	20.7	
04-OCT-76	11:55:19	-5	32.35	0.0	36.3	2.53	0.0	9.8	
04-OCT-76	11:56:15	-4	32.32	0.0	36.3	2.60	0.0	6.8	
04-OCT-76	11:57:11	-3	32.32	0.0	36.3	2.64	0.0	7.6	
04-OCT-76	11:58:07	-2	32.35	0.0	36.4	2.62	0.0	10.9	
04-OCT-76	11:59:04	-1	32.38	0.0	36.5	2.68	0.0	17.2	
04-OCT-76	12:00:02	0	32.38	0.0	36.5	5.57	0.0	20.0	

FIGURE 11.

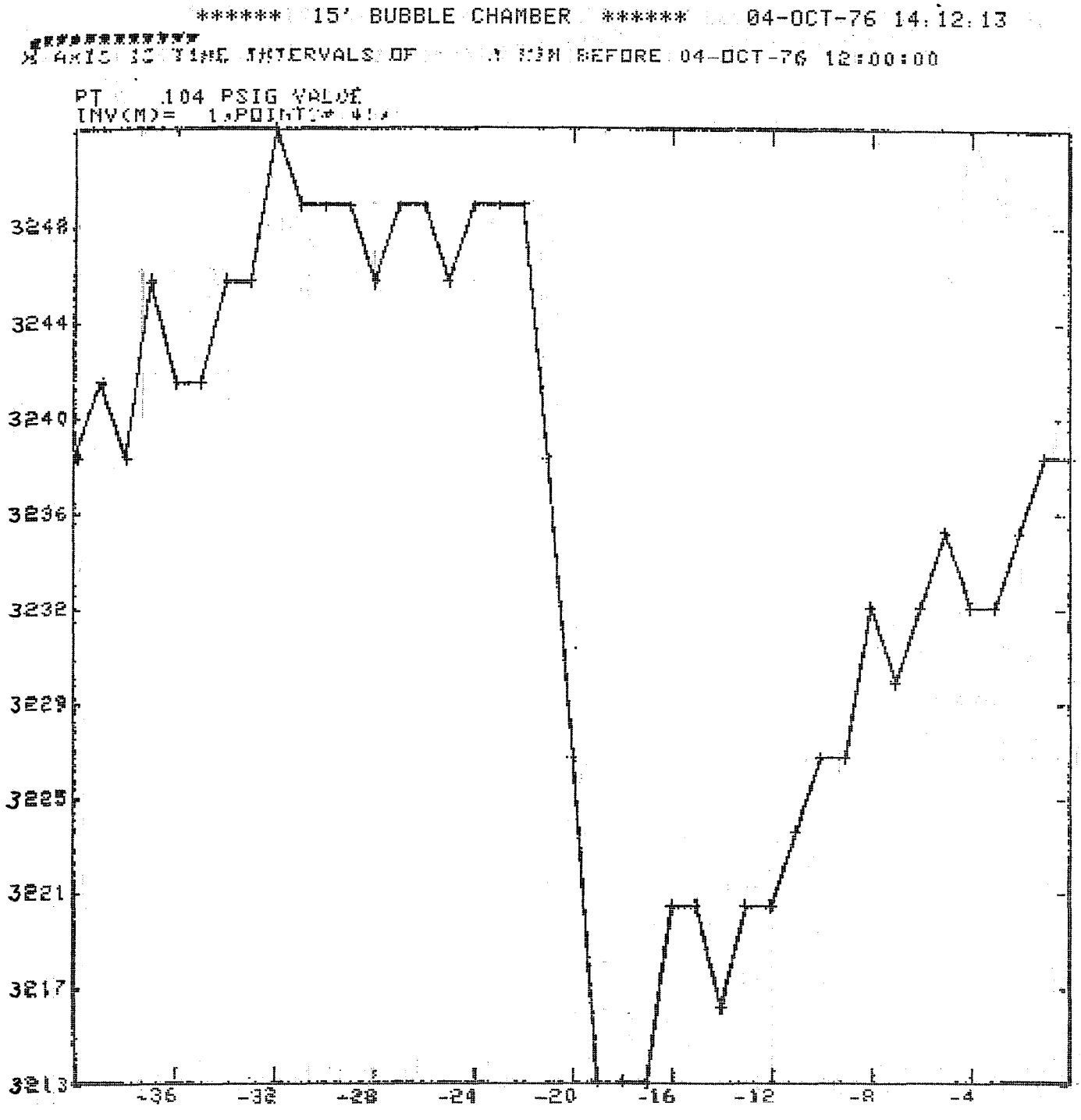

****** 15' BUBBLE CHAMBER ****** 04-OCT-76 14:12:13
X AXIS IS TIME INTERVALS OF .1 MIN BEFORE 04-OCT-76 12:00:00
PT .104 PSIG VALUE
INV(M)= 1,POINTS* 4!*
3248
3244
3240
3236
3232
3229
3225
3221
3217
3213
-36
-32
-28
-24
-20
-16
-12
-8
-4

FIGURE 12.

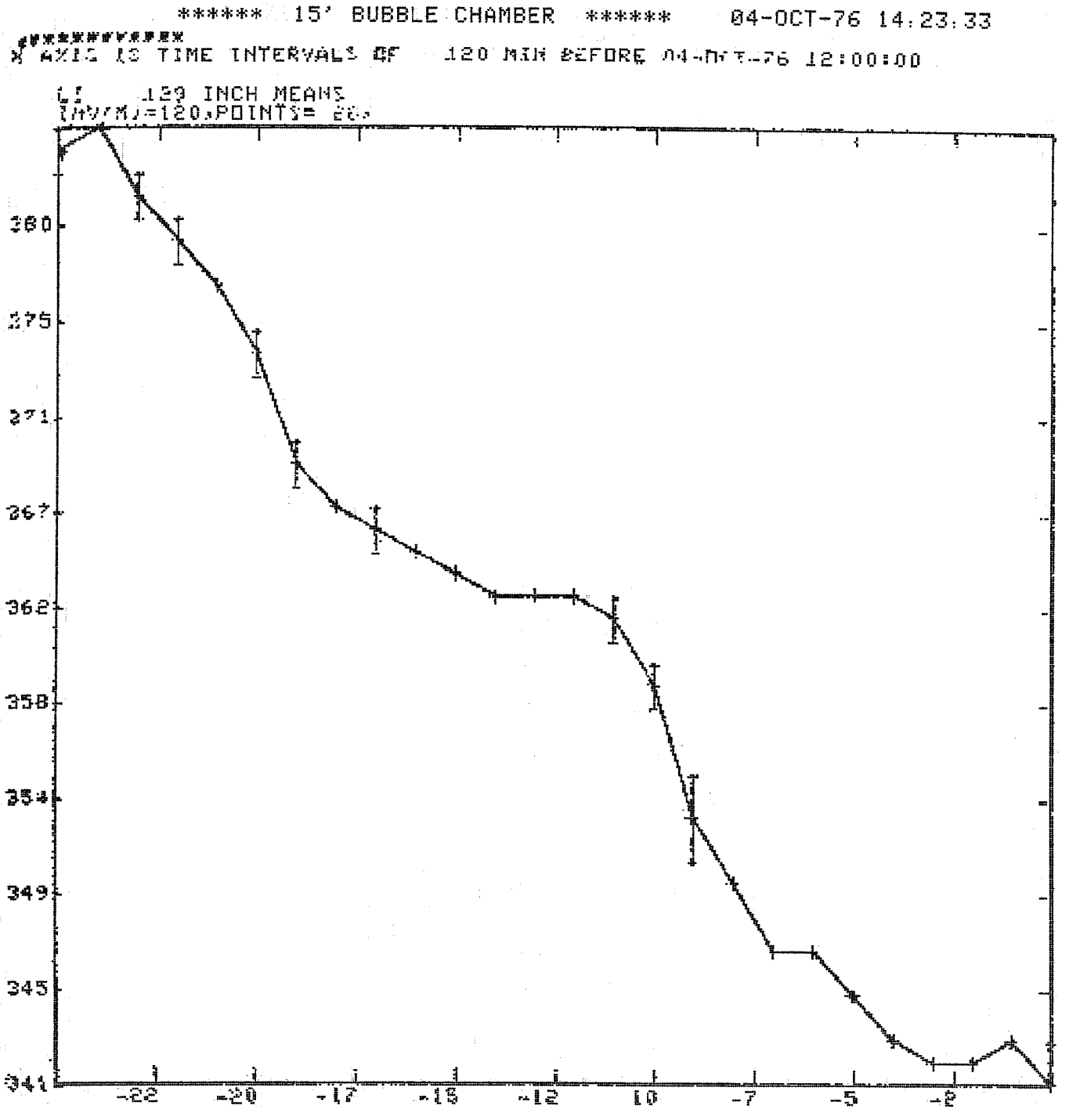

****** 15' BUBBLE CHAMBER ******
04-OCT-76 14:23:33
X AXIS IS TIME INTERVALS OF 120 MIN BEFORE 04-OCT-76 12:00:00
139 INCH MEANS
380
375
371
367
362
358
354
349
345
341
-22
-20
-17
-15
-12
-10
-7
-5
-2

FIGURE 13

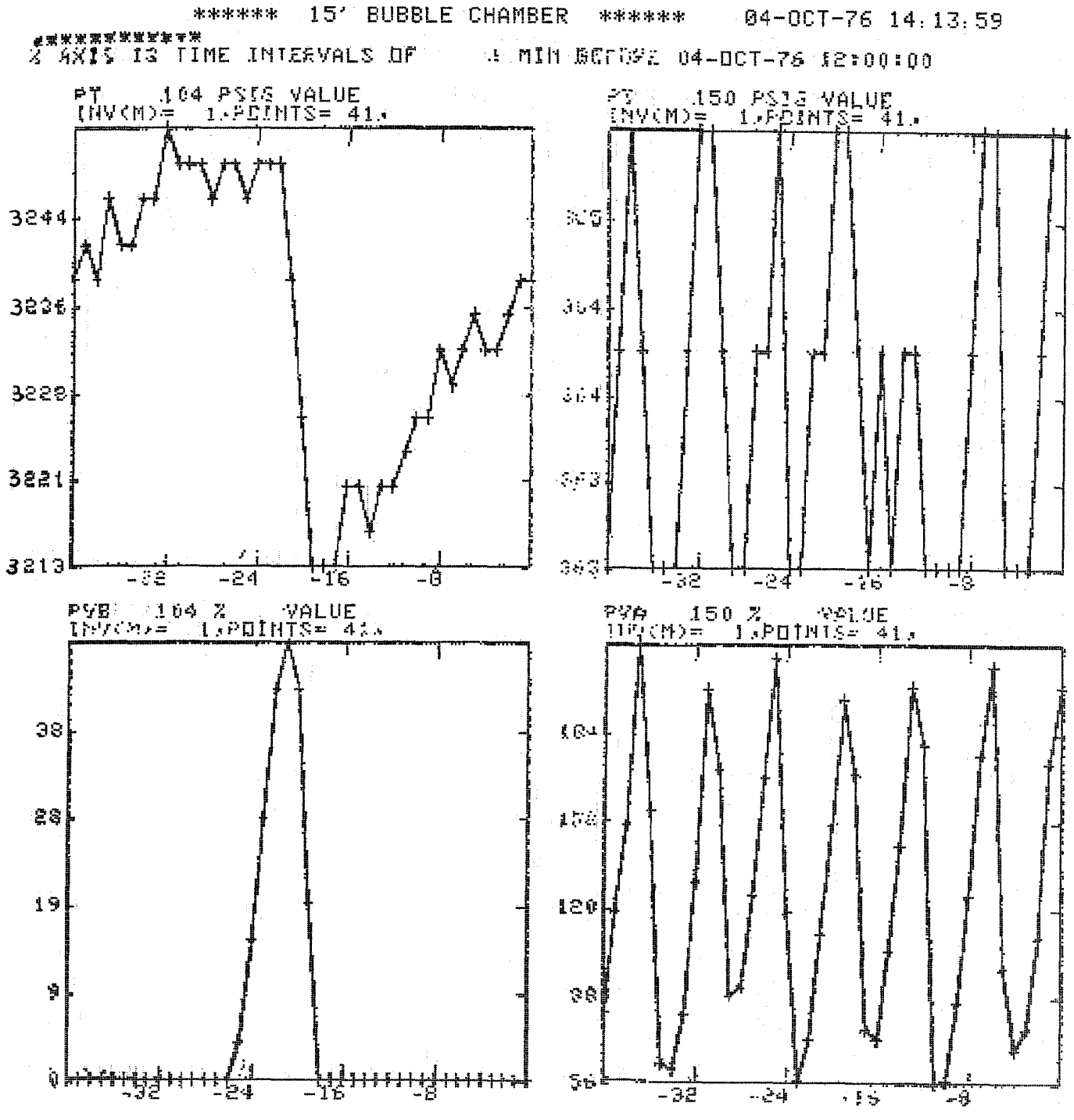

****** 15' BUBBLE CHAMBER ******
04-OCT-76 14:13:59
X AXIS IS TIME INTERVALS OF
MIN BEFORE 04-OCT-76 12:00:00
PT .104 PSIG VALUE
INV(M)= 1.POINTS= 41.
3244
3236
3228
3221
3213
-32
-24
-16
-8
PT .150 PSIG VALUE
INV(M)= 1.POINTS= 41.
PVB .104 % VALUE
INV(M)= 1.POINTS= 41.
38
28
19
9
0
PVA .150 % VALUE
INV(M)= 1.POINTS= 41.

FIGURE 14.

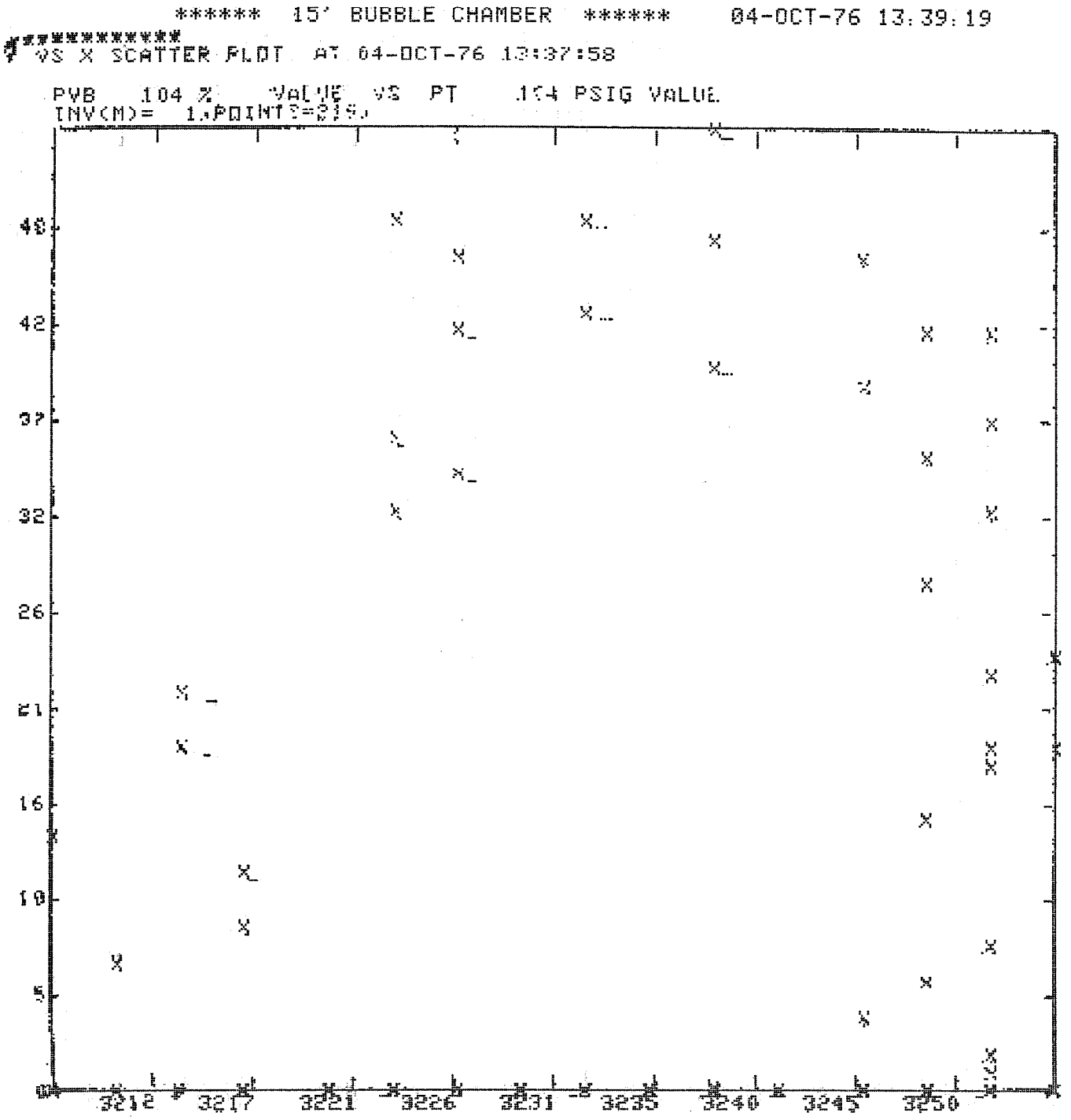

****** 15' BUBBLE CHAMBER ****** 04-OCT-76 13:39:19
VS X SCATTER PLOT AT 04-OCT-76 13:37:58
PVB 104 % VALVE VS PT 104 PSIG VALVE
INV(M)= 1,POINTS=215.
48
42
37
32
26
21
16
10
5
0
3212
3217
3221
3226
3231
3235
3240
3245
3250

FIGURE 15.

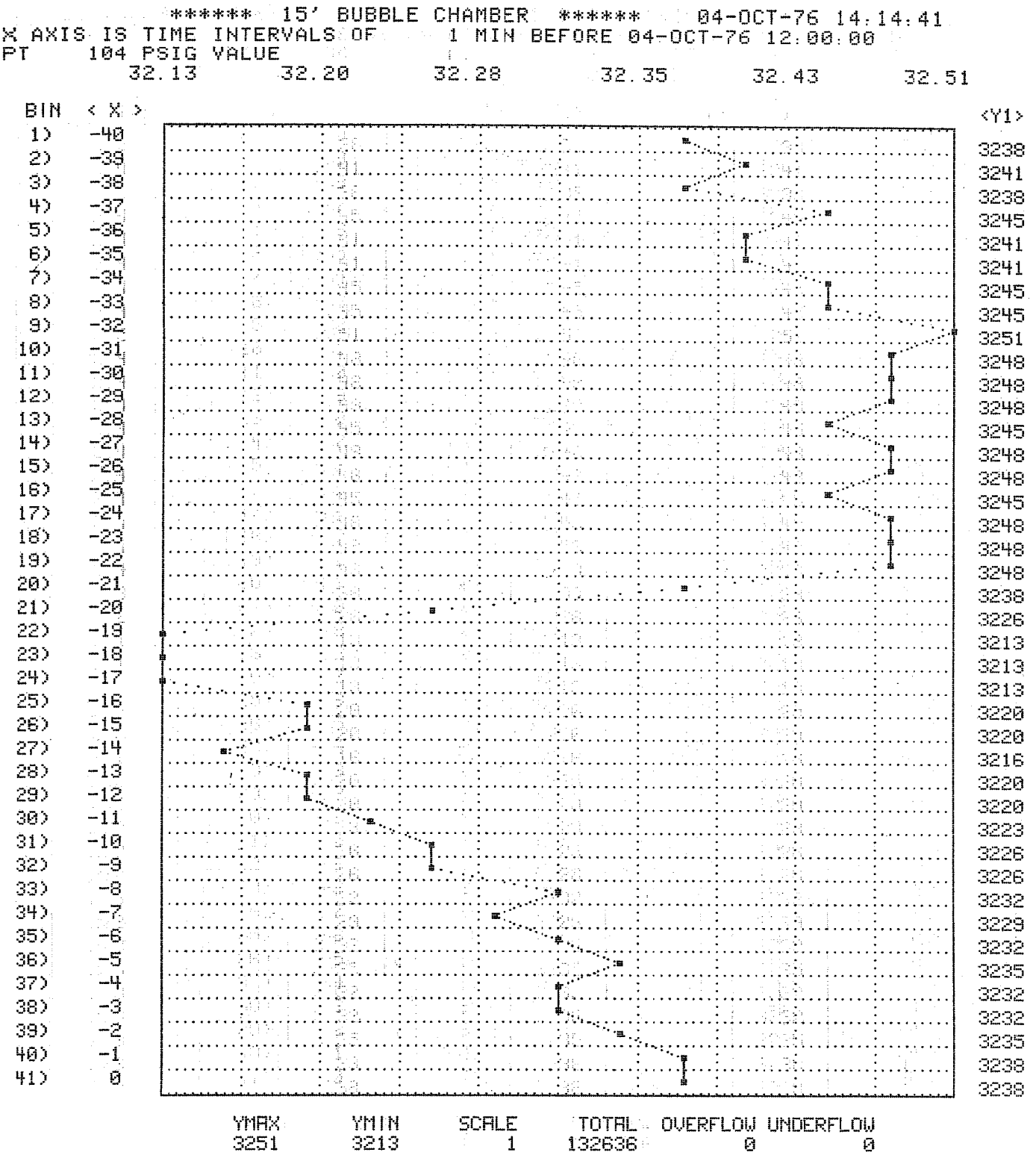

****** 15' BUBBLE CHAMBER ****** 04-OCT-76 14:14:41
X AXIS IS TIME INTERVALS OF 1 MIN BEFORE 04-OCT-76 12:00:00
PT 104 PSIG VALUE
32.13 32.20 32.28 32.35 32.43 32.51
BIN < X >
<Y1>
1) -40 3238
2) -39 3241
3) -38 3238
4) -37 3245
5) -36 3241
6) -35 3241
7) -34 3245
8) -33 3245
9) -32 3251
10) -31 3248
11) -30 3248
12) -29 3248
13) -28 3245
14) -27 3248
15) -26 3248
16) -25 3245
17) -24 3248
18) -23 3248
19) -22 3248
20) -21 3238
21) -20 3226
22) -19 3213
23) -18 3213
24) -17 3213
25) -16 3220
26) -15 3220
27) -14 3216
28) -13 3220
29) -12 3220
30) -11 3223
31) -10 3226
32) -9 3226
33) -8 3232
34) -7 3229
35) -6 3232
36) -5 3235
37) -4 3232
38) -3 3232
39) -2 3235
40) -1 3238
41) 0 3238
YMAX YMIN SCALE TOTAL OVERFLOW UNDERFLOW
3251 3213 1 132636 0 0

FIGURE 16.

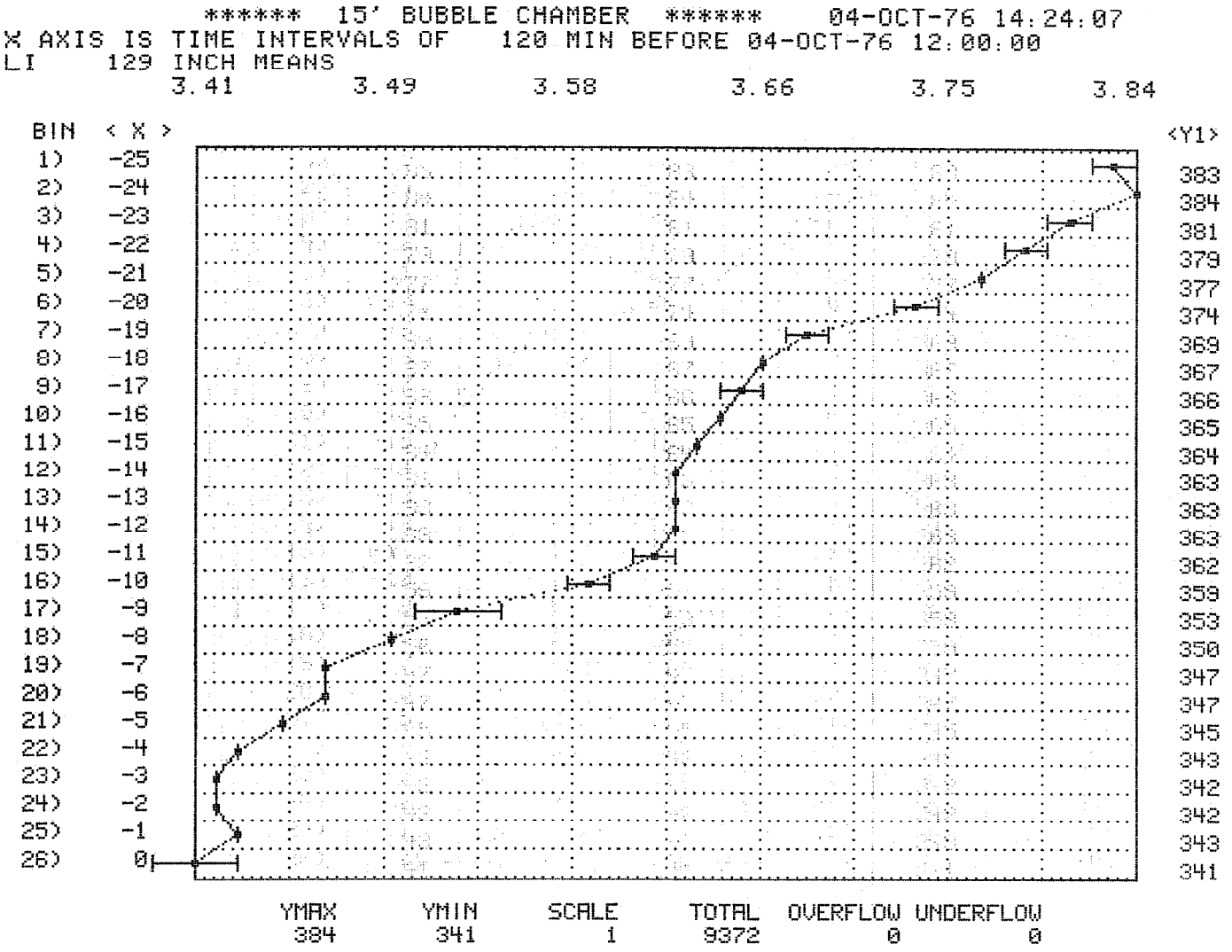

****** 15' BUBBLE CHAMBER ****** 04-OCT-76 14:24:07
X AXIS IS TIME INTERVALS OF 120 MIN BEFORE 04-OCT-76 12:00:00
LI 129 INCH MEANS
3.41 3.49 3.58 3.66 3.75 3.84
BIN < X > <Y1>
1) -25 383
2) -24 384
3) -23 381
4) -22 379
5) -21 377
6) -20 374
7) -19 369
8) -18 367
9) -17 366
10) -16 365
11) -15 364
12) -14 363
13) -13 363
14) -12 363
15) -11 362
16) -10 359
17) -9 353
18) -8 350
19) -7 347
20) -6 347
21) -5 345
22) -4 343
23) -3 342
24) -2 342
25) -1 343
26) 0 341
YMAX YMIN SCALE TOTAL OVERFLOW UNDERFLOW
384 341 1 9372 0 0

FIGURE 17.